

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORPORATION,

Petitioner,

v.

DIRECTSTREAM, LLC,

Patent Owner.

IPR2018-01605, IPR2018-1606, IPR2018-01607
Patent 7,620,900 B2

**PATENT OWNER DIRECTSTREAM, LLC'S
DEMONSTRATIVE EXHIBITS FOR ORAL ARGUMENT**

Pursuant to 37 C.F.R. §42.70(b) and the Board's Order (Paper 57) in the above-referenced proceeding, Patent Owner DirectStream, LLC files its demonstrative exhibits for the oral hearing scheduled for February 4, 2020.

Dated: January 30, 2020

Respectfully submitted,

/Alfonso Chan/

Alfonso Chan, Reg. No. 45,964

achan@shorechan.com

Joseph F. DePumpo, Reg. No. 38,124

jdepumpo@shorechan.com

SHORE CHAN DEPUMPO LLP

901 Main Street, Suite 330

Dallas, Texas 75202

Tel: (214) 593-9110

Fax: (214) 593-9111

Sean Hsu, Reg. No. 69,477

shsu@jvllp.com

Rajkumar Vinnakota*

kvinnakota@jvllp.com

G. Donald Puckett*

dpuckett@jvllp.com

JANIK VINNAKOTA LLP

8111 Lyndon B. Johnson Frwy., #790

Dallas, Texas 75251

Tel: (214) 390-9999

Fax: (214) 888-0219

* Admitted *Pro Hac Vice*

Attorneys for Patent Owner

DirectStream, LLC

CERTIFICATE OF SERVICE

Pursuant to 37.C.F.R. §§42.6(e)(4) and 42.25(b), the undersigned certifies that on January 30, 2020, a complete copy of the foregoing document was filed electronically through the Patent Trial and Appeal Board's PTAB E2E System and provided, via electronic service, to the Petitioner by serving the correspondence address of record as follows:

Joseph A. Micallef
jmicallef@sidley.com
Scott M. Border
sborder@sidley.com
SIDLEY AUSTIN LLP
1501 K. Street N.W.
Washington, DC 20005

Jason P. Greenhut
jgreenhut@sidley.com
SIDLEY AUSTIN LLP
1 South Dearborn
Chicago, IL 60603

/Alfonso Chan/
Alfonso Chan
Reg. No. 45,964
Tel: (214) 593-9110

Patent Owner's Demonstratives

February 4, 2020

Microsoft Corporation, Petitioner,
v.
DirectStream, LLC, Patent Owner

Case IPR2018-01601, -01602, -01603 (Patent No. 7,225,324 B2)
Case IPR2018-01605, 01606, -01607 (Patent No. 7,680,800 B2)

Summary of Argument

TABLE OF CONTENTS

I. INTRODUCTION	1
II. TECHNICAL BACKGROUND OF THE '324 PATENT	3
A. Need for Cost-Effective HPC	4
B. Direct Stream's HPC Advancements	8
1. Architectural Innovations	9
2. Invention of MAP Technology	10
3. DirectStream's Continued Improvement	11
4. DirectStream's Continued Success	14
C. The '324 Patent: SRC Invents Methods for Enhancing Parallelism and Performance in Reconfigurable Computing Systems	15
III. PETITIONER'S EXPERT TESTIMONY IS CONCLUSORY AND IS NOT RELIABLE TO EXPLAIN THE TEACHINGS OF THE PRIOR ART OR SUPPORT A FINDING OF OBVIOUSNESS	17
A. Dr. Stone Fails to Understand the Teachings of the Patent and the Prior Art	17
B. Dr. Stone's Opinions are Grounded in Hindsight Bias	18
C. Dr. Stone Fails to Provide Facts and Data to Support His Opinions	22
IV. PERSON OF ORDINARY SKILL IN THE ART	29
V. DIRECTSTREAM'S CLAIM CONSTRUCTIONS	29
A. "seamless" vs. "systolic" and "data driven"	33
1. "pass computed data seamlessly between said computational loops"	34
2. Systolic and Data Driven	42
B. "stream communication"	50

ii

1. Petitioner's Construction is Deeply Flawed and Illogical Under BRI	50
2. DirectStream's Construction is Reasonable and Consistent with the Plain and Ordinary Meaning	52
C. "computational loop" ... [wherein only functional units needed to solve the calculation are instantiated]	69
VI. THE CHALLENGED CLAIMS ARE PATENTABLE OVER THE PRIOR ART	74
A. Petitioner's Burden of Proof to Invalidate the Patent	74
1. Anticipation	74
2. Obviousness	76
B. The Challenged Claims are Patentable over Splash2	77
1. The Prior Art Does Not Disclose Stream Communication	77
a) Splash2 Does Not Disclose Stream Communication	77
b) Based on its Petition, Petitioner Cannot Now Argue Splash2 Discloses Stream Communication, as Properly Construed	79
c) Petitioner's Other Prior Art Do Not Disclose Stream	80
2. The Prior Art Does Not Disclose Two Computational Loops	80
a) Splash2 Does Not Disclose Two Computational Loops	80
b) Splash2 Also Does Not Disclose Instantiating the Two Computational Loops in the FPGAs	87
c) Petitioner's Other Prior Art Do Not Disclose Two Computational Loops	90

iii

3. The Prior Art Does Not Unambiguously Disclose Seamlessly Passing Computed Data Between Computational Loops	96
a) Splash2 Does Not Unambiguously Disclose Seamlessly Passing Computed Data Between Computational Loops	96
b) Using Petitioner's Own Construction, Splash2 Would Still Fail to Invalidate the '324 Patent	100
c) RaPiD Confirms the Prior Art Teaches Using Memory Between Processing Elements to Store Results	105
d) Roccatano also Cannot Disclose "Seamless" Because it Requires Multiple Processors	106
e) Petitioner's Other Prior Art Just as Ambiguous as Splash2	107
C. Claims are not rendered obvious by Combining Prior Art	109
1. Prior Art Combinations Still Missing Claim Elements	110
2. A POSITA Would Not Have Been Motivated to Combine the Prior Art	111
3. Petitioner and its Expert Failed to Consider Whether it Would Be Feasible to Modify the Teachings of the Prior Art to Combine with Each Other	113
4. Petitioner and its Expert Improperly Rely on Hindsight Reasoning to Combine Prior Art	119
D. The Objective Indicia in this Case Indicate Nonobviousness	121
VII. CONCLUSION	125
VIII. LIST OF EXHIBITS	128

iv

Summary of Argument

TABLE OF CONTENTS

I. PETITIONER PURPOSEFULLY AVOIDS THE TECHNICAL BACKGROUND OF THE '324 PATENT	1
A. The '324 Patent: Methods for Enhancing Parallelism and Performance in Reconfigurable Computing Systems, Using FPGAs.	1
B. Patent Owner's Experts Provide the Board the True State of the Art and Opinions Consistent With An Actual Methodology.	2
C. Petitioner and its Expert Provide No Evidence or Support for Their Understanding of the State of the Art or What a POSITA Would Have Been Motivated to Do In Order to Meet Their Burden of Proof.	4
II. PETITIONER'S EXPERT'S REPLY TESTIMONY AND EVIDENCE IS CONCLUSORY, UNTIMELY, AND NOT RELIABLE.....	6
A. Dr. Stone Fails to Provide Facts and Data to Support His Opinions and Purposefully Misconstrues Patent Owner's Expert Testimony.....	6
B. Petitioner's Reply and New Testimony is Outside the Scope of Patent Owner's Response or Should Have Been Provided in His Original Declaration.	8
III. PATENT OWNER'S CLAIM CONSTRUCTIONS.....	9
A. "stream communication"	10
B. "computational loop" [wherein only functional units needed to solve the calculation are instantiated].....	15
1. Computational Loops Are Not Infinite Loops.....	15
2. Looping controlled by the Sun workstation.....	17
C. "pass computed data seamlessly"	20
D. "seamless" vs. "systolic" and "data driven".....	21
IV. OBJECTIONS TO THESE PROCEEDINGS.....	21

Institution Grounds

Case Number(s)	Reference(s)	Basis	Claim(s) Challenged
IPR2018-01601, IPR2018-01602, IPR2018-01603	Splash2	35 U.S.C. §§ 102(a) and 102(b) ⁴	1, 15, 18, 21, and 22
IPR2018-01601, IPR2018-01602, IPR2018-01603	Splash2	35 U.S.C. § 103(a)	1, 15, 18, 21, and 22
IPR2018-01601, IPR2018-01602, IPR2018-01603	Splash2 and Gaudiot	35 U.S.C. § 103(a)	1, 15, 18, 21, and 22
IPR2018-01601	Splash2 and RaPiD	35 U.S.C. § 103(a)	8 and 9

Institution Grounds

Case Number(s)	Reference(s)	Basis	Claim(s) Challenged
IPR2018-01601	Splash2, RaPiD, and Gaudiot ⁵	35 U.S.C. § 103(a)	8 and 9
IPR2018-01601	Splash2 and Jeong	35 U.S.C. § 103(a)	20
IPR2018-01601	Splash2, Jeong, and Gaudiot	35 U.S.C. § 103(a)	20
IPR2018-01602	Splash2 and Chunky SLD	35 U.S.C. § 103(a)	7, 17, and 24
IPR2018-01602	Splash2, Chunky SLD, and Gaudiot	35 U.S.C. § 103(a)	7, 17, and 24
IPR2018-01603	Splash2 and Roccatano	35 U.S.C. § 103(a)	2–5, 22, and 23
IPR2018-01603	Splash2, Roccatano, and Gaudiot	35 U.S.C. § 103(a)	2–5, 22, and 23

1601 vs 1605 Cross-Reference

- 1601 and 1605 Petitions pertain to related patents
- The asserted prior art is the same across both consolidated proceedings

PO Exhibit List Description	1601 Ex. No.	1605 Ex. No.
Deposition Transcript of Harold Stone dated 5/30/19	2066	2065
Stone 1987 - HPC Architecture	2070	2069
Deposition Transcript of Stephen Trimmerger dated 6/7/19	2076	2075
U.S. Patent 6,339,819 B1	2085	2084
Declaration of Jon Huppenthal dated 7/11/19	2100	2101
SRC Carte TMC Programming Environment v3.0 Guide (Pre-Release)	2107	2108
Declaration of Dr. Houman Homayoun dated 7/25/19	2111	2112
[28] DirectHit SEC Filings, located at https://www.sec.gov/Archives/edgar/data/1092756/0000912057-99-010346.txt	2139	2140
[44] U.S. Patent No. 8,589,666	2155	2156
Declaration of Tarek El-Ghazawi dated 7/23/19	2164	2166
[EL-GH08] Tarek El-Ghazawi, Esam El-Araby, Miaoqing Huang, Kris Gaj, Volodymyr Kindratenko, and Duncan Buell, "The Promise of High-Performance Reconfigurable Computing," IEEE Computer, vol. 41, no. 2, pp. 69-76, February 2008	2165	2167
[BUEL07] Buell, El-Ghazawi, Gaj, and Kindratenko, "High-Performance Reconfigurable Computing" IEEE Computer (Guest Editors Intro), March 2007 (Vol. 40, No. 3).	2166	2168
[1005] Halverson, "The Functional Memory Approach to the Design of Custom Computing Machines," Dissertation University of Hawaii, August 1994	2167	2169
U.S. Patent No. 5,748,613	2169	2171
European Patent EP 1 820 309 B1	2170	2172
U.S. Patent No. 8,543,746	2171	2173
U.S. Patent No. 8,352,456	2172	2174
U.S. Patent Pub. No. 2010/0070730 A1	2173	2175
Deposition Transcript of Dr. Harold S. Stone dated December 13, 2019	2176	2178
Supplemental Declaration of Dr. Houman Homayoun Under 37 CFR §42.64(B)(2)	2177	2179

Burden of Proof for Invalidity

Legal Authority - Burden of Proof in IPRs

- Petitioner bears the burden of proving by a preponderance of the evidence, with substantial evidence, that the Patent is invalid under 35 U.S.C. §§102 and 103
- *See Corning Inc. v. DSM LP Assets B.V*, IPR2013-00048, Paper 96 at 4 (P.T.A.B. July 11, 2014) (emphasis in original) (“Showing a reasonable likelihood of prevailing [for institution] is less stringent a standard than prevailing by a preponderance of the evidence.”).

Requirements for Anticipation

- Under 35 U.S.C. §102, a claim is “anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference.” *Verdegaal Bros. v. Union Oil Co. of Cal.*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987).

No ambiguity

- Claim elements must be described in a single reference with “sufficient precision and detail to establish that the subject matter existed in the prior art.” *Verve, LLC v. Crane Cams, Inc.*, 311 F.3d 1116, 1120 (Fed. Cir. 2002); *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989).
- Ambiguous references do not anticipate claims. *Wasica Fin. GmbH v. Cont’l Auto. Sys., Inc.*, 853 F.3d 1272, 1284 (Fed. Cir. 2017) (finding claim not invalidated because prior art was ambiguous on whether requisite disclosure would be present to a POSITA and therefore did not anticipate the claim).

Requirements for Obviousness

- Obviousness requires:
 - (1) “all the claimed elements were known in the prior art,”
 - (2) “one skilled in the art could have combined the elements as claimed by known methods with no change in their respective functions,” and
 - (3) “the combination yielded nothing more than predictable results to one of ordinary skill in the art.”
- MPEP §2143(A) (emphasis added) (citing *KSR*, 550 U.S. at 416; *Sakraida v. AG Pro, Inc.*, 425 U.S. 273, 282 (1976); *Anderson’s-Black Rock, Inc. v. Pavement Salvage Co.*, 396 U.S. 57, 62-63 (1969); *Great Atl. & P. Tea Co. v. Supermarket Equip. Corp.*, 340 U.S. 147, 152 (1950)).

Requirements for Obviousness

- “An invention is not obvious simply because all of the claimed limitations were known in the prior art at the time of the invention. Instead, we ask ‘whether there is a reason, suggestion, or motivation in the prior art that would lead one of ordinary skill in the art to combine the references, and that would also suggest a reasonable likelihood of success.’” *Caterpillar Inc.*, IPR2017-02188, Paper 71 at 17 (Final Written Decision) (quoting *Forest Labs, LLC v. Sigmapharm Labs., LLC*, 918 F.3d 928, 934 (Fed. Cir. 2019); *Smiths Indus. Med. Sys., Inc. v. Vital Signs, Inc.*, 183 F.3d 1347, 1356 (Fed. Cir. 1999)).

Requirements for Obviousness

- In evaluating combinations of the prior art, it is not sufficient to say that a result may occur from a given set of conditions, but rather, it must occur. *PersonalWeb Techs., LLC. v. Apple, Inc.*, 917 F.3d 1376, 1382 (Fed. Cir. 2019). If an equally plausible or more plausible interpretation of the prior art can be supported by evidence, then obviousness cannot be found through an application of inherency. *Id.*

Requirements for Obviousness

- Additionally, “it can be important to identify a reason that would have prompted a person of ordinary skill in the relevant field to combine the elements in the way the claimed new invention does.” *KSR*, 550 U.S. at 418; see *also* MPEP §2143(A).
- And, the burden remains on Petitioner to demonstrate “what [skilled artisans] would have been motivated to do.” *ZTE*, 685 Fed. App’x 939-40.
- “If any of these findings cannot be made, then this rationale cannot be used to support a conclusion that the claim would have been obvious to one of ordinary skill in the art.” MPEP §2143(A).

Rational underpinning

- “[R]ejections on obviousness cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness.” *KSR*, 550 U.S. at 418.
- Additionally, objective evidence relevant to nonobviousness (“secondary considerations”) may include evidence of commercial success, long-felt but unsolved needs, failure of others, and unexpected results. *See Graham*, 383 U.S. at 17-18.

All PO evidence must be considered

- All PO evidence must be considered:

Petitioner's objections are without merit and of the type the Board frowns upon as no analytical attempt was made to actually form a proper evidentiary objection.⁴ It will be reversible error if the Board does choose to ignore or strike all these exhibits. *See, e.g., Knauf Insulation, Inc. v. Rockwool Int'l A/S*, No. 2018-1810, *et al.*, 2019 WL 5152356, at *4 (Fed. Cir. Oct. 15, 2019) (vacating Board invalidity ruling based on improper interpretation of prior art reference at the time of invention); *Personal Web Techs., LLC v. Apple, Inc.*, 917 F.3d 1376, 1382-83 (Fed. Cir. 2019) (same); *Align Tech., Inc. v. ClearCorrect Operating, LLC*, 745 Fed. App'x 361, 364-65 (Fed. Cir. 2018) (same).

Claim Construction Standard

Claim Construction

- Claim construction and determination of claim scope must be proper to evaluate validity
 - Phillips standard – claims given ordinary and customary meaning
 - Phillips standard – intrinsic evidence first
 - Phillips standard – extrinsic evidence if the intrinsic evidence is unclear, but it must still be consistent with intrinsic record
 - Cannot exclude preferred embodiment
 - Claim differentiation, preserve meaning and scope of different claims
 - Separate claim terms should be given separate meaning

Claim Construction

- Phillips standard – claims given ordinary and customary meaning
- The words of a claim should be given their “ordinary and customary meaning,” which is “the meaning that the term[s] would have to a [POSITA]...at the time of the invention.” *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312-13 (Fed. Cir. 2006) (en banc).
- The Board should also consider the context in which the term is used in an asserted claim or in related claims in the patent or specification. *Id.* at 1313


Claim Construction

- Broadest Reasonable Interpretation – construction must still be reasonable
- “The broadest reasonable interpretation does not mean the broadest possible interpretation.” See MPEP §2111. “Rather, the meaning given to a claim term must be consistent with the ordinary and customary meaning of the term (unless the term has been given a special definition in the specification), and must be consistent with the use of the claim term in the specification and drawings.” *Id.*

Patent and Claims

Patent Summary

- EX1001 – US Patent 7,225,324
- EX1005 – US Patent 7,620,800
- The '800 Patent is a continuation of the '324 Patent and both are in the same family



US007225324B2

(12) **United States Patent**
Huppenthal et al.

(10) **Patent No.:** US 7,225,324 B2
(45) **Date of Patent:** May 29, 2007

(54) **MULTI-ADAPTIVE PROCESSING SYSTEMS AND TECHNIQUES FOR ENHANCING PARALLELISM AND PERFORMANCE OF COMPUTATIONAL FUNCTIONS**

(75) **Inventors:** Jan M. Huppenthal, Colorado Springs, CO (US); David E. Caliga, Colorado Springs, CO (US)

(73) **Assignee:** SRC Computers, Inc., Colorado Springs, CO (US)

(*) **Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 550 days.

(21) **Appl. No.:** 10/285,318

(22) **Filed:** Oct. 31, 2002

(65) **Prior Publication Data**
US 2004/0088527 A1 May 6, 2004

(51) **Int. Cl.**
G06F 17/00 (2006 01)

(52) **U.S. Cl.** 712/226

(58) **Field of Classification Search** 712/15, 712/19, 226, 215

See application file for complete search history.

(56) **References Cited**
U.S. PATENT DOCUMENTS

4,727,503 A	2/1988	McWhiter
4,872,133 A *	10/1989	Leeland
4,962,281 A *	10/1990	HeBig, Sr
5,020,059 A	5/1991	Gera et al.
5,072,371 A *	12/1991	Benner et al.
5,220,057 A	7/1993	Slado et al.
5,274,832 A *	12/1993	Khan
5,471,627 A	11/1995	Means et al.
5,477,221 A	12/1995	Chang et al.

5,570,040 A 10/1996 Lytle et al.

5,640,586 A * 6/1997 Peckanek et al. 712/13

5,737,766 A 4/1998 Jan

5,784,108 A * 7/1998 Skalercky et al. 375/240.15

5,802,962 A 4/1999 Cantner

5,903,771 A 5/1999 Sgro et al.

5,915,123 A * 6/1999 Murky et al. 712/16

5,956,518 A * 9/1999 DeLeon et al. 712/15

6,023,755 A 2/2000 Casselman

6,052,773 A 4/2000 DeLeon et al.

6,061,706 A * 5/2000 Gai et al. 708/491

6,076,152 A 6/2000 Huppenthal et al.

6,192,439 B1 2/2001 Grasso et al.

6,215,508 B1 * 4/2001 Woodfill et al. 382/154

6,226,776 B1 5/2001 Paschall et al.

6,289,440 B1 * 6/2001 Casselman

6,385,757 B1 * 5/2002 Gupta et al. 716/1

OTHER PUBLICATIONS

Rosenberg, J.M. Dictionary of Computers, Information Processing & Telecommunications, 1984, John Wiley & Sons, 2ed, pp. 496.*

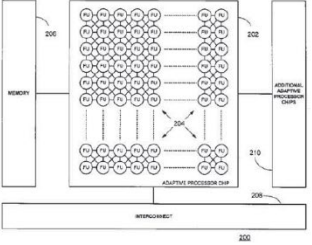
(Continued)

Primary Examiner—Eric Coleman
(74) *Attorney, Agent, or Firm*—William J. Kubisa; Michael C. Martensen; Hogan & Hartson LLP

(57) **ABSTRACT**

Multi-adaptive processing systems and techniques for enhancing parallelism and performance of computational functions are disclosed which can be employed in a myriad of applications including multi-dimensional pipeline computations for seismic applications, search algorithms, information security, chemical and biological applications, filtering and the like as well as for systolic waveform computations for fluid flow and structures analysis, bioinformatics etc. Some applications may also employ both the multi-dimensional pipeline and systolic waveform methodologies disclosed.

52 Claims, 20 Drawing Sheets



Petitioner Microsoft Corporation - Ex. 1001, p. 1

Purpose of the patent

- The '324 patent claims techniques for enhancing parallelism and performance in reconfigurable computing systems. EX1001,1:37-41.
- At the time of the invention, “most large software applications achieve[d] high performance operation through the use of parallel processing” that required “multiple processors to work simultaneously on the same problem.” EX1001, 1:42-50.

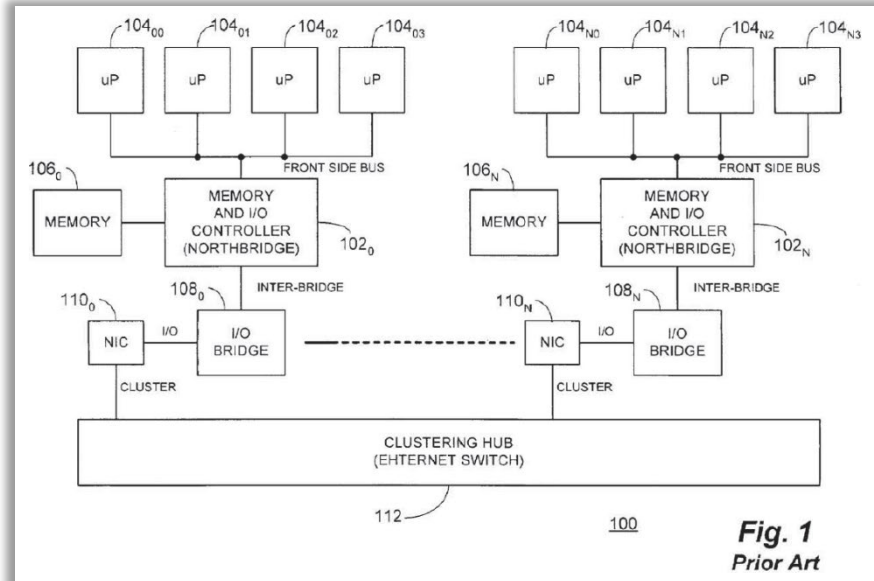


Fig. 1
Prior Art

Purpose of the patent

- The specification discusses the problem of passing data over numerous boundaries (or seams) between processing elements in typical multi-processor systems.
 - "In a multi-processor, microprocessor-based system, each processor is allocated but a relatively small portion of the total problem called a cell. However, to solve the total problem, results of one processor are often required by many adjacent cells because their cells interact at the boundary and upwards of six or more cells, all having to interact to compute results, would not be uncommon. Consequently, intermediate results must be passed around the system in order to complete the computation of the total problem. This, of necessity, involves numerous other chips and busses that run at much slower speeds than the microprocessor thus resulting in system performance often many orders of magnitude lower than the raw computation time."
- EX1005 at 2:26-38.

Purpose of the patent

- The problem was that “as more and more performance is required, so is more parallelism, resulting in ever larger systems” to the point that “[c]lusters exist ... that have tens of thousands of processors and can occupy football fields of space.” EX1001, 1:50-56. “Systems of such a large physical size present many obvious downsides, including, among other factors, facility requirements, power, heat generation and reliability.” EX1001, 1:56-59.

Purpose of the patent

- In a multi-processor, microprocessor-based system, each processor is allocated but a relatively small portion of the total problem called a cell. However, to solve the total problem, results of one processor are often required by many adjacent cells because their cells interact at the boundary and upwards of six or more cells, all having to interact to compute results, would not be uncommon. Consequently, intermediate results must be passed around the system in order to complete the computation of the total problem. This, of necessity, involves numerous other chips and busses that run at much slower speeds than the microprocessor thus resulting in system performance often many orders of magnitude lower than the raw computation time.
EX1001, 2:25-37 (emphasis added).

Purpose of the patent

- The inventors of the '324 patent realized that this problem could be solved by “a processor technology ... that offers orders of magnitude more parallelism per processor.” EX1001, 1:63-65.
- And that this type of processor technology is “possible through the use of a reconfigurable processor” because reconfigurable processors can “instantiate as many functional units as may be required to solve the problem up to the total capacity of the integrated circuit chips they employ.” EX1001, 1:65-2:5.
- The inventors of the '324 patent also realized that additional, and less obvious, performance gains could “also be realized by reconfigurable processors due to the much tighter coupling of the parallel functional units within each chip than can be accomplished in a microprocessor-based computing system.” EX1001, 2:17-24.

Purpose of the patent

- In a reconfigurable computing system, “since ten to one thousand times more computations can be performed within a single chip, any boundary data that is shared between these functional units need never leave a single integrated circuit chip.” EX1001, 2:38-42 (emphasis added).
- “Therefore, data moving around the system, and its impact on reducing overall system performance, can also be reduced by two or three orders of magnitude.” EX1001, 2:42-45.

Purpose of the patent

(12) **United States Patent**
Huppenthal et al.

(54) **MULTI-ADAPTIVE PROCESSING SYSTEMS AND TECHNIQUES FOR ENHANCING PARALLELISM AND PERFORMANCE OF COMPUTATIONAL FUNCTIONS**

(75) Inventors: **Jon M. Huppenthal**, Colorado Springs, CO (US); **David E. Caliga**, Colorado Springs, CO (US)

(73) Assignee: **SBC Computers, Inc.**, Colorado Springs, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 550 days.

(21) Appl. No.: 10/285,318

(22) Filed: **Oct. 31, 2002**

(65) **Prior Publication Data**
US 2004/0088527 A1 May 6, 2004

(51) **Int. Cl.**
G06F 17/00 (2006.01)

(52) **U.S. Cl.** 712/226

(58) **Field of Classification Search** 712/15,

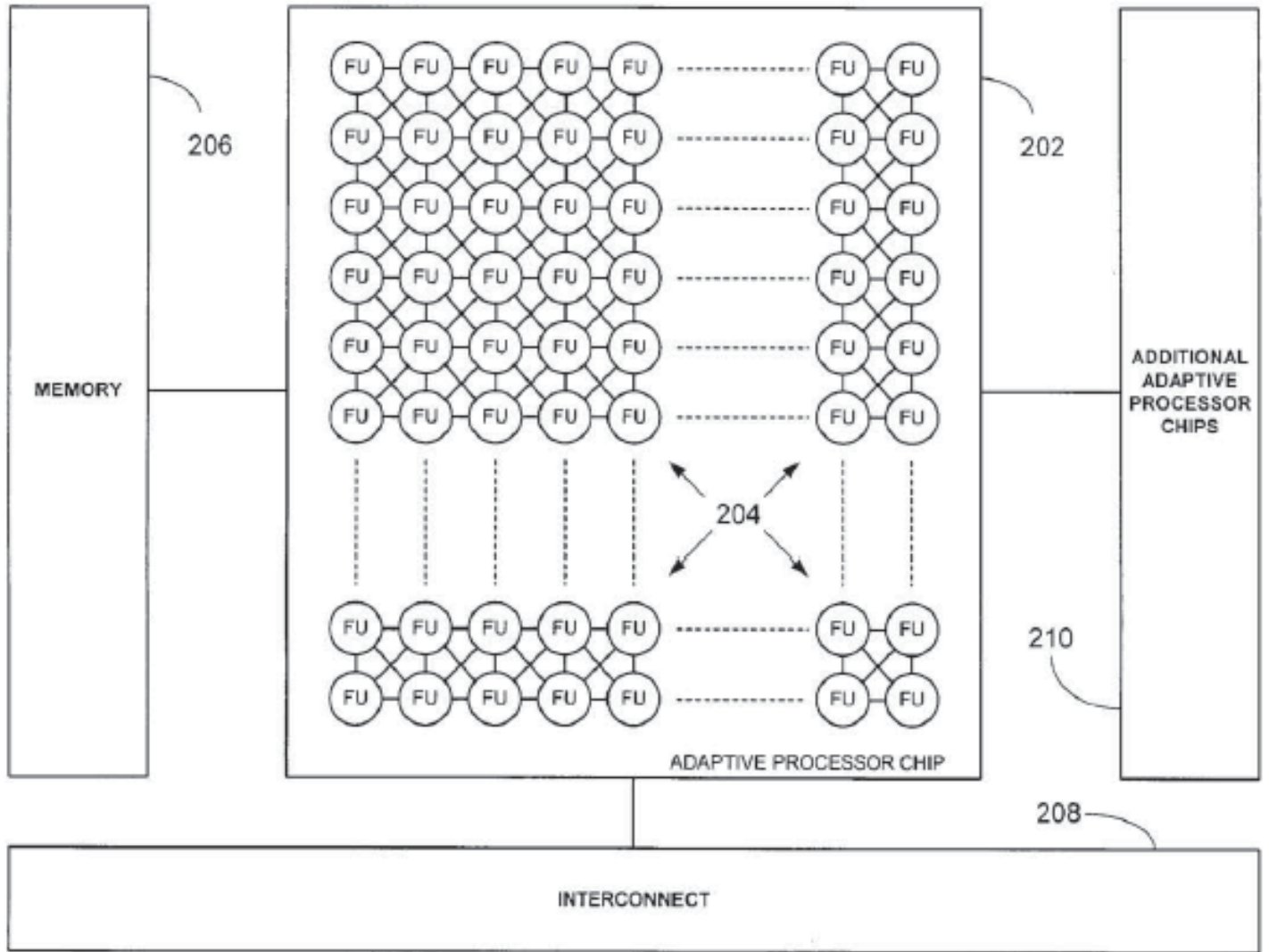
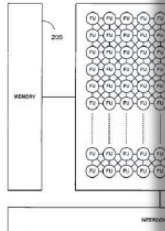
712/19, 226, 215

See application file for complete search history.

(56) **References Cited**

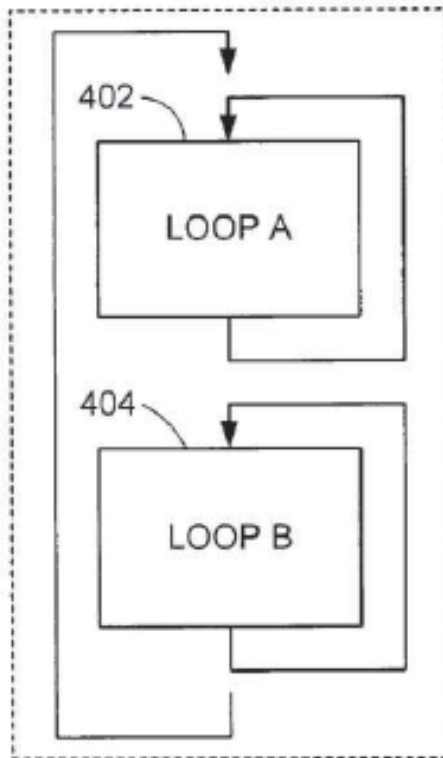
U.S. PATENT DOCUMENTS

4,727,503 A	2/1988	McWhirter	
4,872,133 A *	10/1989	Lecland	708/509
4,962,381 A *	10/1990	Helbig, Sr.	342/372
5,030,659 A	5/1991	Costa et al.	
5,072,371 A *	12/1991	Benner et al.	712/11
5,230,657 A	7/1993	Shido et al.	
5,276,832 A *	12/1993	Khan	708/434
5,471,627 A	11/1995	Menas et al.	
5,477,221 A	12/1995	Chang et al.	



200 **Fig. 2**

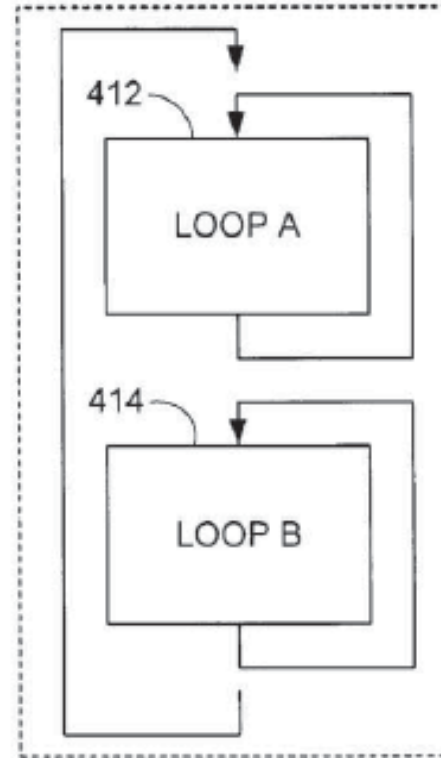
Purpose of the patent



PHASE 1
- LOOP A "ACTIVE"
- LOOP B "INACTIVE"

PHASE 2
- LOOP A "INACTIVE"
- LOOP B "ACTIVE"

Fig. 4A
Prior Art

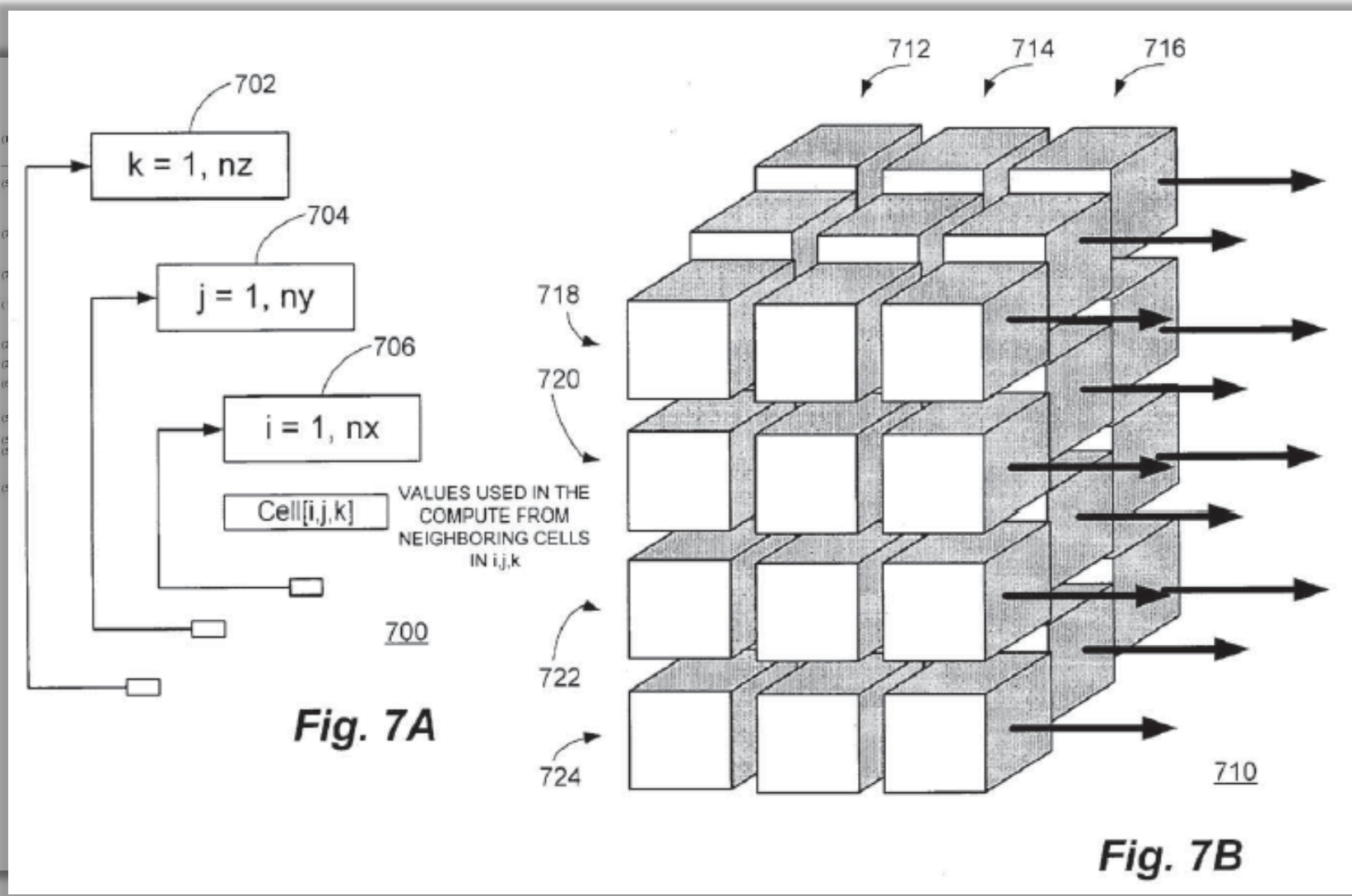


PHASE 1
- LOOP A WORKS ON
DIMENSION 1
- LOOP B WORKS ON
DIMENSION 0
(may be dummy data)

PHASE 2
- LOOP A WORKS ON
DIMENSION 2
- LOOP B WORKS ON
DIMENSION 1

Fig. 4B

Purpose of the patent



Purpose of the '324 Patent – Independent claim 1



(12) United States Patent (10) Patent No.: US 7,225,324 B2

The invention claimed is:

55 1. A method for data processing in a reconfigurable computing system, the reconfigurable computing system comprising at least one reconfigurable processor, the reconfigurable processor comprising a plurality of functional units, said method comprising:

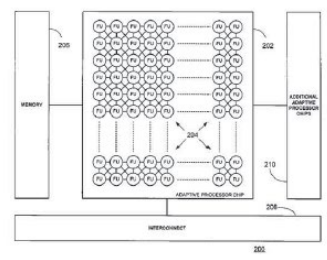
60 transforming an algorithm into a calculation that is systolically implemented by said reconfigurable computing system at the at least one reconfigurable processor;

instantiating at least two of said functional units at the at least one reconfigurable processor to perform said calculation wherein only functional units needed to solve the calculation are instantiated and wherein each

65 instantiated functional unit at the at least one reconfigurable

urable processor interconnects with each other instantiated functional unit at the at least one reconfigurable processor based on reconfigurable routing resources within the at least one reconfigurable processor as established at instantiation, and wherein systolically 5 linked lines of code of said calculation are instantiated as clusters of functional units within the at least one reconfigurable processor;

utilizing a first of said instantiated functional units to operate upon a subsequent data dimension of said 10 calculation forming a first computational loop; and substantially concurrently utilizing a second of said instantiated functional units to operate upon a previous data dimension of said calculation forming a second 15 computational loop wherein said systolic implementation of said calculation enables said first computational loop and said second computational loop execute concurrently and pass computed data seamlessly between said computational loops.



Petitioner Microsoft Corporation - Ex. 1001, p. 1

Purpose of the '324 Patent – Dependent claim 15



(12) **United States Patent**
Huppenthal et al.
 (10) **Patent No.:** US 7,225,324 B2
 (45) **Date of Patent:** May 29, 2007

(54) **MULTI-ADAPTIVE PROCESSING SYSTEMS AND TECHNIQUES FOR ENHANCING PARALLELISM AND PERFORMANCE OF COMPUTATIONAL FUNCTIONS**
 (75) **Inventors:** Jon M. Huppenthal, Colorado Springs, CO (US); David E. Caliga, Colorado Springs, CO (US)
 (73) **Assignee:** SRC Computers, Inc., Colorado Springs, CO (US)
 (*) **Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 550 days.

5,570,640 A 10/1996 Lytle et al.
 5,640,536 A * 01/1997 Pechanek et al.
 5,737,766 A 4/1998 Iain
 5,784,108 A * 7/1998 Skaletzky et al.
 5,892,962 A 4/1999 Cloutier
 5,903,771 A 5/1999 Sgro et al.
 5,915,123 A * 6/1999 Minsky et al.
 5,956,518 A * 9/1999 DeHon et al.
 6,025,755 A 2/2000 Casselman
 6,052,773 A 4/2000 DeHon et al.
 6,061,796 A * 5/2000 Gai et al.
 6,076,152 A 6/2000 Hagooshah et al.
 6,192,439 B1 2/2001 Grunewald et al.
 6,215,898 B1 * 4/2001 Woodfill et al. 382/154
 6,236,776 B1 5/2001 Bushall et al.
 6,289,440 B1 * 9/2001 Casselman 712/227
 6,385,757 B1 * 5/2002 Gupta et al. 716/1

15. The method of claim 1 wherein instantiating includes establishing a stream communication connection between functional units.

55

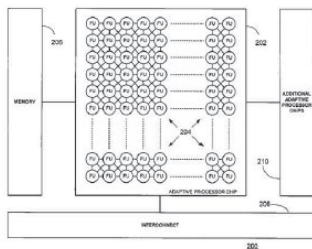
(21) **Appl. No.:** 10/285,318
 (22) **Filed:** Oct. 31, 2002
 (65) **Other Publications**
 US 2004/0088527 A1 May 6, 2004
 (51) **Int. Cl.**
G06F 17/00 (2006.01)
 (52) **U.S. Cl.** 712/226
 (58) **Field of Classification Search** 712/15, 712/19, 226, 215
 See application file for complete search history.

OTHER PUBLICATIONS
 Rosenberg, J.M., Dictionary of Computers, Information Processing & Telecommunications, 1984, John Wiley&Sons, 2nd, pp. 496-*

(56) **References Cited**
U.S. PATENT DOCUMENTS
 4,727,503 A 2/1988 McWhirter
 4,872,133 A * 10/1989 Lecland 708/509
 4,962,381 A * 10/1990 Heibig, Sr 342/372
 5,030,659 A 5/1991 Costa et al.
 5,072,371 A * 12/1991 Benner et al. 712/11
 5,230,657 A 7/1993 Shido et al.
 5,274,832 A * 12/1993 Khan 708/434
 5,471,627 A 11/1995 Meaus et al.
 5,477,221 A 12/1995 Chang et al.

ABSTRACT
 Multi-adaptive processing systems and techniques for enhancing parallelism and performance of computational functions are disclosed which can be employed in a myriad of applications including multi-dimensional pipeline computations for seismic applications, search algorithms, information security, chemical and biological applications, filtering and the like as well as for systolic wavefront computations for fluid flow and structures analysis, bioinformatics etc. Some applications may also employ both the multi-dimensional pipeline and systolic wavefront methodologies disclosed.

52 Claims, 20 Drawing Sheets



Petitioner Microsoft Corporation - Ex. 1001, p. 1

Purpose of the '800 Patent – Independent claim 1

- systolic / data driven
- instantiated / formed



(12) **United States Patent**
Huppenthal et al.
 (10) **Patent No.:** US 7,620,800 B2
 (45) **Date of Patent:** *Nov. 17, 2009

(54) **MULTI-ADAPTIVE PROCESSING SYSTEMS AND TECHNIQUES FOR ENHANCING PARALLELISM AND PERFORMANCE OF COMPUTATIONAL FUNCTIONS**
 (75) **Inventors:** Jan M. Huppenthal, Colorado Springs, CO (US); David E. Calliga, Colorado Springs, CO (US)
 (73) **Assignee:** SRC Computers, Inc., Colorado Springs, CO (US)
 (*) **Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 95 days.
 This patent is subject to a terminal disclaimer.

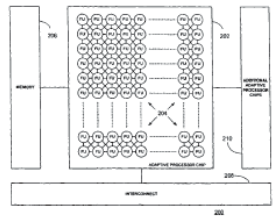
(21) **Appl. No.:** 11/733,064
 (22) **Filed:** Apr. 9, 2007
 (65) **Prior Publication Data**
 US 2007/0204131 A1 Aug. 30, 2007
Related U.S. Application Data
 (63) Continuation of application No. 10/285,318, filed on Oct. 31, 2002, now Pat. No. 7,225,324.

(51) **Int. Cl.**
G06F 15/82 (2006.01)
 (52) **U.S. Cl.** 712/226
 (58) **Field of Classification Search** 712/226, 712/15, 19, 215
 See application file for complete search history.
 (56) **References Cited**
 U.S. PATENT DOCUMENTS
 4,727,503 A 2/1988 McWhitter
 4,763,294 A 8/1988 Fong
 4,872,133 A 10/1989 Leeland
 4,962,381 A 10/1990 Helbig, Sr.

5,820,889 A 5/1991 Corin et al.
 5,072,371 A 12/1991 Benner et al.
 5,230,057 A 7/1993 Shido et al.
 5,274,832 A 12/1993 Khan
 5,471,627 A 11/1995 Means et al.
 5,477,221 A 12/1995 Chang et al.
 (Continued)
FOREIGN PATENT DOCUMENTS
 JP 59-206972 11/1984
 (Continued)
OTHER PUBLICATIONS
 Gandiot, Jean-Luc, Data-Driven Multicomputers in Digital Signal Processing, 1987, IEEE, Proceedings of the IEEE, vol. 75, No. 9, pp. 1229-1234.*
 (Continued)
Primary Examiner—Eric Coleman
 (74) **Attorney, Agent, or Firm**—Michael C. Martensen; William J. Kubisa; Hogan & Hartson LLP
 (57) **ABSTRACT**

Multi-adaptive processing systems and techniques for enhancing parallelism and performance of computational functions are disclosed which can be employed in a myriad of applications including multi-dimensional pipeline computations for seismic applications, search algorithms, information security, chemical and biological applications, filtering and the like as well as for systolic waveform computations for fluid flow and structures analysis, bioinformatics etc. Some applications may also employ both the multi-dimensional pipeline and systolic waveform methodologies disclosed.

52 Claims, 20 Drawing Sheets



Petitioner Microsoft Corporation - Ex. 1005, p. 1

Disputed Claim Terms

Disputed Claim Terms

- “pass computed data seamlessly between said computational loops”
- “systolic” and “data driven”
- “computational loop”
- “stream communication”

“pass computed data seamlessly between said computational loops”

1. “pass computed data seamlessly between said computational loops”

DirectStream’s Construction	Petitioner’s Construction
communicating the computed data over the reconfigurable routing resources	to communicate computed data directly

“pass computed data seamlessly between said computational loops”

- Petitioner’s construction of this term improperly introduces the limitation of “directly” that is not supportable by the intrinsic or extrinsic evidence
- DirectStream’s proposed construction comes directly from the intrinsic record and captures the plain and customary understanding that “seamless” should be without seams or boundaries between processing elements. EX2111 ¶¶ 159-168, 220-223..

“pass computed data seamlessly between said computational loops”

- Dr. Stone testified that the word “directly” means “the data goes from the first to the second without going to something intervening.” EX2064 at 85:14- 24; EX2111¶171.
- But when questioned what constitutes “intervening structures” Dr. Stone was unable to specifically identify anything because “I think you’re opening a whole universe.” EX2064 at 86:13-18; EX2111¶172.

13 Q. Okay. Are there any other examples
14 of intervening structures or circuits that
15 would violate this direct connection?

16 A. I -- I think you're opening a
17 universe. I'm not going to answer that because
18 I'd like to -- let's get specific things.

“pass computed data seamlessly between said computational loops”

- His answers depended on where its expert draws the boundaries of the processing element. EX2064 at 85:25-87:24; EX2111¶173.

Page 85

1 STONE, Ph.D.
2 Oh, there it is. Okay.
3 So this concept of systolic, what's
4 your opinion of what that means?
5 A. My opinion of what it means is in
6 the report. It means it's the characteristic
7 of rhythmically computing and passing data
8 directly between processing elements. And then
9 I quote: "Without a program counter or clock
10 that drives a movement of data," and also
11 operating in a manner that is, "transport
12 triggered, i.e., by the arrival of a data
13 object."
14 Q. Okay. And you mention the word
15 "directly," it was passing data directly
16 between processing elements. What does that
17 phrase mean to you or what's the context?
18 What are you trying to describe
19 there?
20 A. That the data goes from first to the
21 second without going to something intervening.
22 It directly go -- is connected immediately.
23 Indirectly we -- you go through one or more
24 intervening places to get there.
25 Q. Okay. So would memory, if the data

Page 86

1 STONE, Ph.D.
2 was going from one processing element to memory
3 and then back to a processing element, is that
4 something you would consider as an intervening
5 thing?
6 A. Well, that would not be a direct
7 connection of the output of the cell to the
8 next cell. It says, "Between processing
9 elements you're directly connected." If you're
10 saying you have a processing element outputting
11 to memory and then coming back to another
12 processing element, that would not be direct.
13 Q. Okay. Are there any other examples
14 of intervening structures or circuits that
15 would violate this direct connection?
16 A. I -- I think you're opening a
17 universe. I'm not going to answer that because
18 I'd like to -- let's get specific things.
19 Q. Well, how about a -- a register?
20 Would that be an intervening structure?
21 A. I -- I'm puzzled because that --
22 that register would be within -- within the
23 processing element in my mind.
24 Q. Okay.
25 A. If it's within the processing

Page 87

1 STONE, Ph.D.
2 element as a register, yeah, I would put it
3 there, then the output of that register, if
4 it's connected directly to the input of the
5 next processing element, would be direct.
6 Q. Okay. So your view of the register
7 would be that it's part of the processing
8 element and, therefore, is not intervening is
9 your opinion?
10 MR. MICALLEF: Objection. Outside
11 the scope.
12 THE WITNESS: When you rephrased it,
13 I'm not sure I got all the words, all the
14 ifs, ands and buts.
15 BY MR. HSU:
16 Q. Let me try again. So your view of
17 the register would be it's part of the
18 processing element and, therefore, is not
19 intervening?
20 MR. MICALLEF: Same objection.
21 Excuse me.
22 THE WITNESS: I'm -- see, I didn't
23 say that. If the register is part of the
24 processing element, then the connection
25 would be direct. And I don't know exactly

“pass computed data seamlessly between said computational loops”

- Petitioner’s construction depends on where its expert draws the boundaries of the processing element. EX2064 at 88:12-91:24; EX2111¶¶174-175.

1 STONE, Ph.D.
2 the motivation for leaving it out of the
3 processing element, so I'm confused there
4 as to what it would be doing there.

5 I -- I'd have to look at the -- at
6 the cells, the processing elements, and I
7 would be able to look and see, is the
8 output directly connected to the next
9 input. That's how I would make my
10 judgment.

11 BY MR. HSU:

12 Q. What about a buffer? Is that
13 something that you would consider an
14 intervening structure?

15 MR. MICALLEF: Objection. Outside
16 the scope.

17 THE WITNESS: Well, the buffer is
18 more complex. Is it itself a processing
19 element? Then it's directly connected from
20 a processing element to the processing
21 element buffer, and from there, directly
22 connected to the next processing element.

23 I'd have to see the context of this
24 to see if it -- if the connections are
25 direct, because as you describe it, it's

16 Q. Yeah, let me try that again because

17 I know I stumbled all over stuff too.

18 So in this hypothetical that you
19 kind of outlined, there was a Processing
20 Element 1 and then a Processing Element 3, and
21 we're trying to send data from 1 to 3. If
22 there's a Processing Element 2 in between,
23 would that constitute an intervening structure?

24 MR. MICALLEF: Objection. Outside
25 the scope. Lack of foundation.

1 STONE, Ph.D.

2 THE WITNESS: Again, I would have to
3 look at the whole thing, but the way you
4 described it, we have direct connections 1
5 to 2, 2 to 3, and if it's rhythmically
6 computing and passing data directly between
7 processing elements, it would still be
8 systolic. So I don't know why it wouldn't
9 be systolic.

19 Q. And if Processing Element 2 had a
20 buffer in there, does it change the analysis at
21 all?

22 MR. MICALLEF: Objection. Outside
23 the scope. Lack of foundation.

24 THE WITNESS: So within Processing
25 Element 2, there's a buffer and there's

1 STONE, Ph.D.

2 boundaries of Processing Element 2, and at
3 the boundaries of Processing Element 2 it
4 is directly connected from the output of 1
5 into the input of 3. So it's still -- all
6 the connections that you describe are still
7 direct connections.

8 BY MR. HSU:

9 Q. Well, I'm wondering, earlier you
10 said that you have a direct connection from 1
11 to 2 and then from 2 to 3, but not a direct
12 connection from 1 directly to 3, and I'm
13 wondering if you introduce a buffer in
14 Processing Element 2, which I think was your
15 original hypothetical, I was asking does that
16 change that direct connection between 1 and 3
17 or the lack thereof?

18 MR. MICALLEF: Same objections.

19 THE WITNESS: The inclusion of the
20 buffer doesn't change anything because the
21 direct connections only are with respect to
22 the boundaries of the processing element,
23 so whatever is inside 2 doesn't change my
24 opinion on what's direct.

25 BY MR. HSU:

“pass computed data seamlessly between said computational loops”

- Petitioner argues in reply:
 - DirectStream also erroneously asserts Dr. Stone testified that if a register were between processing elements there could still be a direct connection between those processing elements.
Response, 41, citing EX2064 at 86:19-88:10, 88:12-91:24. That’s not what he said. In the cited testimony, ***Dr. Stone stated he was talking about a register that was “within” a processing element, not one that was between processing elements.*** See EX2064, 86:21-87:5 (“A. I -- I'm puzzled because that -- that register would be within -- within the processing element in my mind. Q. Okay. A. ***If it's within the processing element as a register, yeah, I would put it there,*** then the output of that register, if it's connected directly to the input of the next processing element, would be direct.”)
- Reply, 24-25 (emphasis added). .

“pass computed data seamlessly between said computational loops”

- This is the precise problem. See Response, 41. If Dr. Stone deems the register to be “within,” then it must be direct; otherwise if he deems the register to be without, then it is not direct.
- The same circuit would be both direct and indirect, depending on where the boundaries of the “processing element” are arbitrarily drawn with respect to intervening structures, which Dr. Stone concedes he could not clarify because it “open[s] a whole universe.”
- This is not a reasonable claim construction position for Petitioner to take under either Phillips or BRI

“pass computed data seamlessly between said computational loops”

- Petitioner’s inclusion of this extraneous word into the construction does nothing but improperly introduce ambiguity and confusion. EX2111¶¶169-176.
- The ambiguity arising from Petitioner’s insertion of the word “directly” would be avoided by simply specifying that the computed data is communicated over the reconfigurable routing resources on the chip, which all of the experts and the named inventor concur is what the patent teaches.
 - Response, 36 (Dr. Stone EX2064 at 85:14-86:12, 90:19-91:24)
 - Response, 39 (Dr. Homayoun’s report EX2111¶¶161-167, 220-223);
 - Reply, 21 (Mr. Huppenthal’s report, EX2100, 55).

“pass computed data seamlessly between said computational loops”

- The specification discusses the problem of passing data over numerous boundaries (or seams) between processing elements in typical multi-processor systems. EX1005 at 2:26-38.
 - "In a multi-processor, microprocessor-based system, each processor is allocated but a relatively small portion of the total problem called a cell. However, to solve the total problem, results of one processor are often required by many adjacent cells because their cells interact at the boundary and upwards of six or more cells, all having to interact to compute results, would not be uncommon. Consequently, intermediate results must be passed around the system in order to complete the computation of the total problem. This, of necessity, involves numerous other chips and busses that run at much slower speeds than the microprocessor thus resulting in system performance often many orders of magnitude lower than the raw computation time."

“pass computed data seamlessly between said computational loops”

- The specification then discusses how the patent solves this problem by ensuring that “any boundary data” that is shared between processing units “need never leave a single integrated circuit chip.” EX1005 at 2:38-48.
 - On the other hand, in the use of an adaptive processor- based system, since ten to one thousand times more computations can be performed within a single chip, any boundary data that is shared between these functional units need never leave a single integrated circuit chip. Therefore, data moving around the system, and its impact on reducing overall system performance, can also be reduced by two or three orders of magnitude. This will allow both significant improvements in performance in certain applications as well as enabling certain applications to be performed in a practical timeframe that could not previously be accomplished.

“pass computed data seamlessly between said computational loops”

- The specification supports this understanding. The '324 patent describes one of the problems with conventional multi-processor computing systems is that they require intermediate results be passed through numerous chips and busses “that run at much slower speeds than the microprocessors thus resulting in system performance often many orders of magnitude lower than the raw computation time.” EX1001 at 2:25-37, 4:64-5:30.

“pass computed data seamlessly between said computational loops”

- By contrast, the adaptive processor-based system described by the '324 patent can perform “ten to one thousand more computations ... within a single chip” so that “data that is shared between functional need never leave a single integrated circuit chip.” EX1001 at 2:38-48.
- The functional units are interconnected by reconfigurable routing resources. EX1001, Fig. 2, 5:31-51.
- So, any “seamless” on chip communications use the reconfigurable routing resources as opposed to the busses and numerous chips used by conventional multi-processor computing systems

“pass computed data seamlessly between said computational loops”

- File History is consistent with this construction
 - “...more computations can be performed within a single chip and any boundary data that is shared between these functional units need never leave a single integrated circuit chip, eliminating the need for external communication protocols and simplifying internal communications. For example, a compiler associated with the reconfigurable computing system can establish stream connections between functional units that rely on general communication protocols.”
- EX1002 at 117-118.

“pass computed data seamlessly between said computational loops”

- File History is consistent with this construction
 - “Khan and Gupta do not teach performing these calculations in a single processor. Rather multiple processors are disclosed which would require consideration for both internal and external communication protocols.... The invention as claimed states that communication between functional units, and not the processors, is communication protocol independent.... Applicants’ invention utilizes available resources to have an application evaluate a problem in a concurrent data flow sense and not in a pipeline sense. That is, it will “pass” a subsequent dimension of a given problem through a first loop of logic concurrently with the previous dimension of data being processed through a second loop. This type of concurrent operation cannot occur in the pipeline operation described in Khan.”
- EX1002 at 148-150.

“pass computed data seamlessly between said computational loops”

- File History is consistent with this construction
 - Additionally, during prosecution, the applicant argued that the use of the words “protocol independent” in the claims was intended to “impart the ability of the functional units to seamlessly pass computed data between computational loops comprised of functional units.”
- EX1002 at 224.

“pass computed data seamlessly between said computational loops”

- File History is consistent with this construction
 - The applicant explained that “communication between other reconfigurable processors within the system would require communication protocol but communication between functional units within an individual reconfigurable processor is free of such a requirement.”
- EX1002 at 174-75; 224-25.

“pass computed data seamlessly between said computational loops”

- Thus, the prosecution history makes clear that “seamlessly” is achieved by utilizing the reconfigurable routing resources to provide a protocol independent communication without the “seams” typically experienced at the boundary of processors.
EX2111¶¶161-167, 220-223.

“pass computed data seamlessly between said computational loops”

- Petitioner’s construction would also exclude standard FPGAs (including the type described in the embodiments of the ’324 Patent and the specific FPGA chips used in Petitioner’s prior art references) since standard FPGAs contain reconfigurable routing resources (comprising buffers and switches) between the configurable logic blocks.
- For example, the literature on Xilinx FPGA chips shows buffer switch boxes and three-state buffers to connect two or more configurable logic blocks. EX1035 at 31; EX2078 at 19-29, 32-34, 37-41, 46-51, 59-65

“pass computed data seamlessly between said computational loops”

FIELD-PROGRAMMABLE GATE ARRAY TECHNOLOGY

edited by

Stephen M. Trimberger
Xilinx

with contributions by

Stephen M. Trimberger
Xilinx

Dennis McCarty
Telle Whitney
Actel

and
The Technical Staff of Altera Corporation
edited by
Robert Hartmann



SPRINGER SCIENCE+BUSINESS MEDIA, LLC

sharbour@ytlp.com

PATENT OWNER DIRECTSTREAM, LLC
EX. 2078, p. 2

Product Obsolete or Under Obsolescence



XC4000E and XC4000X Series Field Programmable Gate Arrays

May 14, 1999 (Version 1.6)

Product Specification

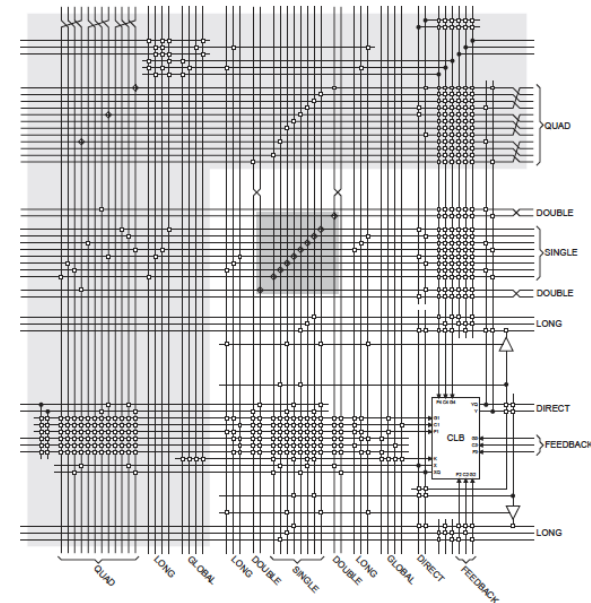
XC4000E and XC4000X Series Features

Note: Information in this data sheet covers the XC4000E, XC4000EX, and XC4000XL families. A separate data sheet covers the XC4000LA and XC4000XV families. Electrical Specifications and package/pin information are covered in separate sections for each family to make the information easier to access, review, and print. For access to these sections, see the Xilinx web site at http://www.xilinx.com/xlnx/web/bil_publications_index.jsp

- System featured Field-Programmable Gate Arrays
 - SelectRAM™ memory, on-chip ultra-fast RAM with synchronous write option
 - dual-port RAM option
 - Fully PCI compliant (speed grades -2 and faster)
 - Abundant flip-flops
 - Flexible function generators
 - Dedicated high-speed carry logic
 - Wide edge decoders on each edge
 - Hierarchy of interconnect lines
 - Internal 3-state bus capability
 - Eight global low-skew clock or signal distribution networks
- System Performance beyond 80 MHz
- Flexible Array Architecture
- Low Power Segmented Routing Architecture
- Systems-Oriented Features
 - IEEE 1149.1-compatible boundary scan logic support
 - Individually programmable output slew rate
 - Programmable input pull-up or pull-down resistors
 - 12 mA sink current per XC4000E output
- Configured by Loading Binary File
 - Unlimited re-programmability
- Read Back Capability
 - Program verification
 - Internal node observability
- Backward Compatible with XC4000 Devices
- Development System runs on most common computer platforms
 - Interfaces to popular design environments
 - Fully automatic mapping, placement and routing
 - Interactive design editor for design optimization

May 14, 1999 (Version 1.6)

Product Obsolete or Under Obsolescence XC4000E and XC4000X Series Field Programmable Gate Arrays



- Common to XC4000E and XC4000X
- XC4000X only
- Programmable Switch Matrix

Figure 27: Detail of Programmable Interconnect Associated with XC4000 Series CLB

6-30

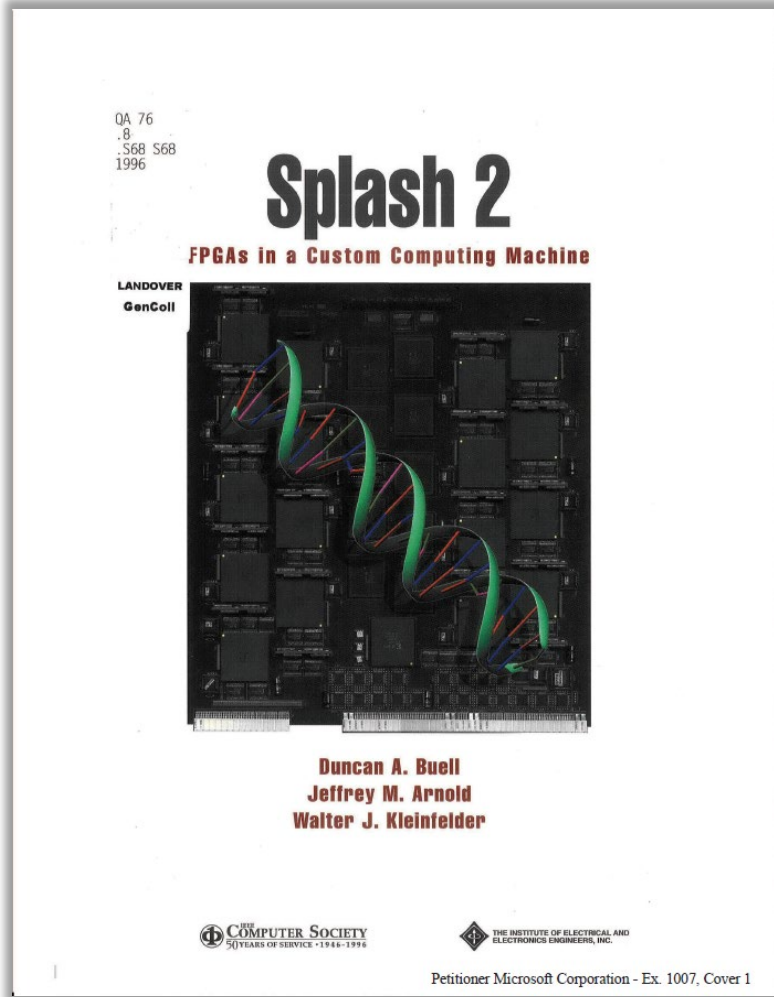
May 14, 1999 (Version 1.6)

Petitioner Microsoft Corporation - Ex. 1035, p. 30

“pass computed data seamlessly between said computational loops”

- Claim differentiation with dependent claims
- Inclusion of “directly” removes instantiation of anything in the reconfigurable routing resources, contrary to plain claim language and dependent claim 15

“pass computed data seamlessly between said computational loops”



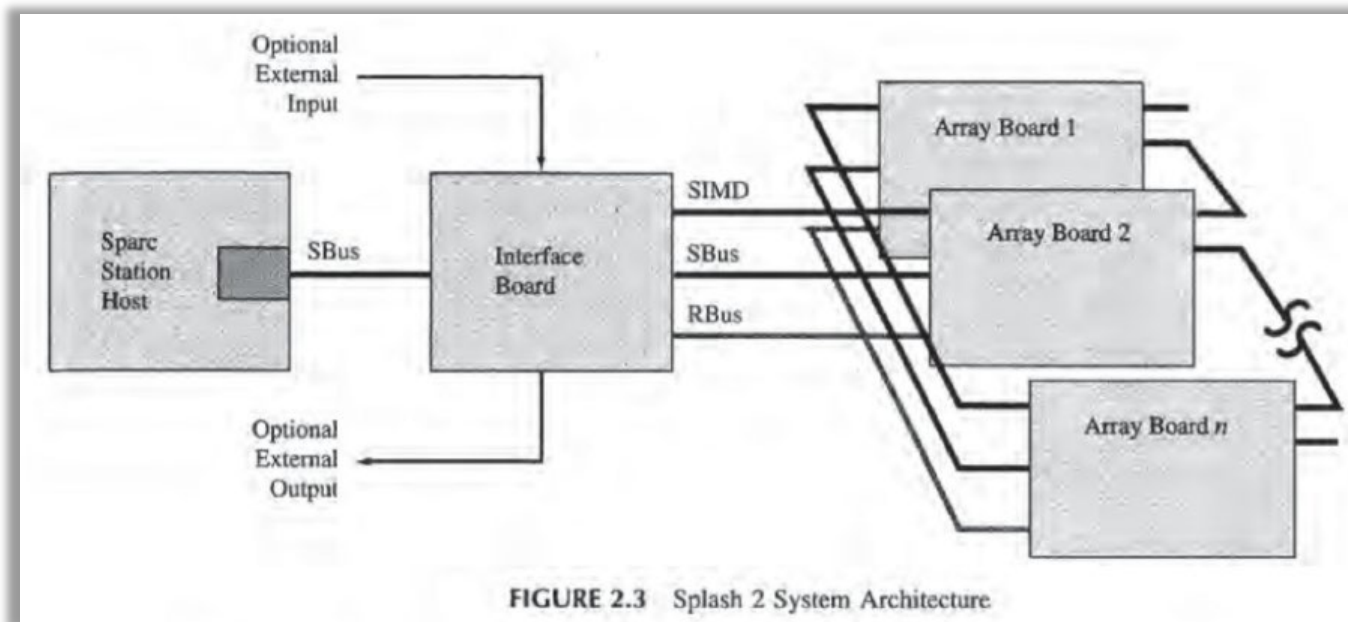
- EX1007 Splash2 prior art
- Passing computed data seamlessly is not taught in Splash2; at best it is ambiguous

“pass computed data seamlessly between said computational loops”

- Splash2’s pseudocode only discloses subroutines that execute once for the current datum to select an execution path for the processor. Thus, they simply are not computational loops.
- Additionally, Splash2 relies on the external Sun workstation to handle any looping, so any computational loop is not even instantiated on the reconfigurable processor. EX2111 ¶209; EX2167 at 14-15; EX2164 ¶¶42-43.

“pass computed data seamlessly between said computational loops”

- The workstation is separate from the array boards containing the FPGAs, EX1007 at 13:



“pass computed data seamlessly between said computational loops”

- ... the FPGAs must communicate with the Sun workstation (which is handling any looping) through the Sbus. EX1007 at 13; EX2111 ¶209; EX2167 at 14-15; EX2164 ¶¶42-43.
- This boundary between the FPGAs and the workstation (through the interface boards) clearly constitutes a “seam” within the context of the '324 Patent and its file history.

“pass computed data seamlessly between said computational loops”

- Petitioner and its expert agree with DirectStream that the claims of the '324 Patent cannot be invalidated by any references that use memory or other structures to provide storage between two processing elements—such an implementation would fail to meet “seamless” limitation of the independent claims.

EX2064 at 85:14-86:12:

EX2064, 91:9-24:

Page 85

1 STONE, Ph.D.
2 Oh, there it is. Okay.
3 So this concept of systolic, what's
4 your opinion of what that means?
5 A. My opinion of what it means is in
6 the report. It means it's the characteristic
7 of rhythmically computing and passing data
8 directly between processing elements. And then
9 I quote: "Without a program counter or clock
10 that drives a movement of data," and also
11 operating in a manner that is, "transport
12 triggered, i.e., by the arrival of a data
13 object."
14 Q. Okay. And you mention the word
15 "directly," it was passing data directly
16 between processing elements. What does that
17 phrase mean to you or what's the context?
18 A. What are you trying to describe
19 there?
20 Q. That the data goes from first to the
21 second without going to something intervening.
22 It directly go -- is connected immediately.
23 Indirectly we -- you go through one or more
24 intervening places to get there.
25 Q. Okay. So would memory, if the data

Page 86

1 STONE, Ph.D.
2 was going from one processing element to memory
3 and then back to a processing element, is that
4 something you would consider as an intervening
5 thing?
6 A. Well, that would not be a direct
7 connection of the output of the cell to the
8 next cell. It says, "Between processing
9 elements you're directly connected." If you're
10 saying you have a processing element outputting
11 to memory and then coming back to another
12 processing element, that would not be direct.
13 Q. Okay. Are there any other examples
14 of intervening structures or circuits that
15 would violate this direct connection?
16 A. I -- I think you're opening a
17 universe. I'm not going to answer that because
18 I'd like to -- let's get specific things.
19 Q. Well, how about a -- a register?
20 Would that be an intervening structure?
21 A. I -- I'm puzzled because that --
22 that register would be within -- within the
23 processing element in my mind.
24 Q. Okay.
25 A. If it's within the processing

Page 91

1 STONE, Ph.D.
2 boundaries of Processing Element 2, and at
3 the boundaries of Processing Element 2 it
4 is directly connected from the output of 1
5 into the input of 3. So it's still -- all
6 the connections that you describe are still
7 direct connections.
8 BY MR. HSU:
9 Q. Well, I'm wondering, earlier you
10 said that you have a direct connection from 1
11 to 2 and then from 2 to 3, but not a direct
12 connection from 1 directly to 3, and I'm
13 wondering if you introduce a buffer in
14 Processing Element 2, which I think was your
15 original hypothetical, I was asking does that
16 change that direct connection between 1 and 3
17 or the lack thereof?
18 MR. MICALLEF: Same objections.
19 THE WITNESS: The inclusion of the
20 buffer doesn't change anything because the
21 direct connections only are with respect to
22 the boundaries of the processing element,
23 so whatever is inside 2 doesn't change my
24 opinion on what's direct.
25 BY MR. HSU:

“pass computed data seamlessly between said computational loops”

- In accordance with the specification and the file history for the '324 Patent, this would certainly include any structures that require data to leave the reconfigurable resources on a single chip for storage and then be read back into the chip by the next processing element.
EX1002 at 117-118, 147-148, 174-175, 224-225.

“pass computed data seamlessly between said computational loops”

- Splash2 is, at best, ambiguous on whether memory is used to store the results from each processing element after each time step to preserve it for output and later use. EX2111 ¶¶210-219. The Splash2 algorithms disclosed indicate storage is likely necessary to preserve the values calculated at each timestep and to store them for some number of additional timesteps. EX2111 ¶¶210-219.
- Based on the disclosed algorithms, Splash2’s pseudocode will overwrite the computed data at each timestep. EX2111 ¶214. Without storage to preserve the computed data at each timestep, intermediate computed data will be lost and the only preserved “computed data” would be the one resulting from the final time step. EX2111 ¶214.

“pass computed data seamlessly between said computational loops”

- In fact, Splash2 clearly discloses providing local memory at each FPGA for storage purposes. EX1007 at 95 (“Many Splash 2 applications use the off-chip memory... which are often used as lookup tables or as storage for results to the host.”),

³In an actual implementation, these two unidirectional distance streams can be combined into one bidirectional stream, using one storage register instead of two. Here we keep the distance streams distinct for clarity.

- EX1007 at 102 (describing the use of one or two storage registers)

- System featured Field-Programmable Gate Arrays
 - SelectRAM™ memory: on-chip ultra-fast RAM with
 - synchronous write option
 - dual-port RAM option

- EX1035 at 1; EX2111¶¶210-219.

“pass computed data seamlessly between said computational loops”

- Other literature about Splash2 confirms this local memory can be used for storage of results. EX2156 at 205-206; EX2111¶¶215-219.

through all the PEs. The PEs can read data either from their respective memory or from any other PE. A broadcast path also exists by suitably programming X_0 .

Splash 2 system supports several models of computation, including PEs executing the same instruction on multiple data (SIMD mode) and PEs executing multiple instructions on multiple data (MIMD mode). It can also execute the same or different instructions on single data by receiving data through the global broadcast bus. The most common mode of operation is systolic in which the SIMD Bus is used for data transfer. Also individual memory available with each PE makes it convenient to store temporary results and tables.

“pass computed data seamlessly between said computational loops”

- Splash2 also discloses using a register for communicating data between processing elements.

7.4.1 Nearest-Neighbor Communication

Left-to-right communication is accomplished with structural connections between virtual processors. Each PE has a left and right port. The width of the communication ports are defined at compile time. These ports are hard-wired together so that the right port of processor i and the left port of processor $i + 1$ share a register. An exception to this is on the Xilinx chip boundaries. The Splash 2 linear interconnect is used for chip-to-chip communication. On each chip, the left port of the first virtual processor on the chip and the right port of the last virtual processor on the chip are connected to XP_LEFT and XP_RIGHT, respectively.

- EX1007 at 88
- The well-known solution at the time of the invention was to use memory storage to smooth out those timing problems, and Splash2 touts its local memory attached to each FPGA as a major benefit for programmers. EX1007 at 13, 40; EX1035 at 1.

“pass computed data seamlessly between said computational loops”

- At best, Splash2 is still ambiguous whether or not it uses the available local memory to store results.
- Here, it is equally (if not more) plausible for a POSITA to interpret Splash2 to use the local memory due to the known timing problems in systolic systems prior to the invention of the '324 Patent. EX2111 ¶¶210-219.

“pass computed data seamlessly between said computational loops”

- Petitioner’s expert even admits that local memory must be used to store temporary results. EX2064 at 176:13-177:25; EX2111 ¶¶215.

Page 176	Page 177
<p>1 STONE, Ph.D. 2 where do those end up going in terms of the 3 bidirectional array? 4 A. They go left and right. 5 Q. Is it a -- a stream of output? 6 A. No, they're used -- look to the 7 upper code again. 8 Do you see that calculation when 9 both of the characters are non null? It's a 10 minimum of three things, one of which is 11 PEDist. That's where it's used. That's how 12 you make the calculation. 13 Q. Okay. I'm going to show you Page 14 102 of Exhibit 1007 and calling your attention 15 to the Footnote 3 at the bottom of the page. 16 Do you have that? 17 A. I see that. 18 Q. What's the one storage register that 19 that footnote is referring to? 20 A. I have to look at the context of 21 this. The two storage registers are the 22 storage registers related to the distance in 23 each character stream. I'm going to read from 24 the text just beneath Figure 8.8. 25 "In addition, there is one distance</p>	<p>1 STONE, Ph.D. 2 stream associated with each character stream." 3 and then the Footnote 3. And the -- the gist 4 of Footnote 3 is we don't really need two 5 distance streams. We can get by with one. 6 Then it makes the additional statement that if 7 you combine the two into one distance stream, 8 we can do that: "...using one storage register 9 instead of two." 10 Now, what happens is when you output 11 something such as TDout or SDout, which appear 12 on Page 101, you have to store them in a 13 register to hold them. This is a matter of how 14 you'd build logic. They're in a register. 15 They hold their value. That value propagates 16 to the next cell, and when the propagation is 17 done, the next cell samples it and then you can 18 clear your register. 19 But there's a hold time and a 20 propagation time that's associated with 21 delivery of information from cell to cell, and 22 that's why you need a storage register. So 23 we've re- -- by reducing two streams -- two 24 distance streams to one, they get by with one 25 storage register.</p>

“pass computed data seamlessly between said computational loops”

- Even under Petitioner’s own proposed construction, Splash2 still does not disclose “seamless” because it cannot show Splash2 discloses passing data “directly.” Petitioner’s own expert testified that “directly” meant nothing can reside in between the boundaries of the two processing elements, including memory, buffers, registers or additional processing elements. EX2064 at 85:14-91:24. Otherwise, it would no longer be direct or seamless.

“pass computed data seamlessly between said computational loops”

- Moreover, Petitioner’s expert did not investigate Splash2 further or either the Xilinx chips or materials on configuring them to better understand the disclosures, even though he admitted he did not have any personal knowledge of them. See EX2064 at 209:2-213:13.
- The Xilinx FPGAs contained in Splash2 clearly contain structure (such as the buffered switch matrix) within the internal routing resources to connect processing elements, which would exclude the Splash2 FPGAs from the definition of Petitioner and its expert.

“pass computed data seamlessly between said computational loops”

- Three-State Buffers

A pair of 3-state buffers is associated with each CLB in the array. (See Figure 27 on page 30.) These 3-state buffers can be used to drive signals onto the nearest horizontal longlines above and below the CLB. They can therefore be used to implement multiplexed or bidirectional buses on the horizontal longlines, saving logic resources. Programmable pullup resistors attached to these longlines help to implement a wide wired-AND function. The buffer enable is an active-High 3-state (i.e. an active-Low enable), as shown in Table 13.

...

Programmable Interconnect

All internal connections are composed of metal segments with programmable switching points and switching matrices to implement the desired routing. A structured, hierarchical matrix of routing resources is provided to achieve efficient automated routing.

- EX1035 at 28-31.

“pass computed data seamlessly between said computational loops”

- Additionally, chapter 2 of the book Field-Programmable Gate Array Technology by Dr. Trimberger describes the Xilinx FPGAs in Splash2, and Dr. Trimberger similarly describes the structures in the routing resources that each add delay to any signals traveling through them, altering the timing of that part of the system.
- EX2078 at 19-29, 32-34, 37-41, 46-51, 59-65, 70.

“pass computed data seamlessly between said computational loops”

FIELD-PROGRAMMABLE GATE ARRAY TECHNOLOGY

edited by

Stephen M. Trimberger
Xilinx

with contributions by

Stephen M. Trimberger
Xilinx

Dennis McCarty
Telle Whitney
Actel

and
The Technical Staff of Altera Corporation
edited by
Robert Hartmann



SPRINGER SCIENCE+BUSINESS MEDIA, LLC

sharbour@ytlp.com

PATENT OWNER DIRECTSTREAM, LLC
EX. 2078, p. 2

Product Obsolete or Under Obsolescence



XC4000E and XC4000X Series Field Programmable Gate Arrays

May 14, 1999 (Version 1.6)

Product Specification

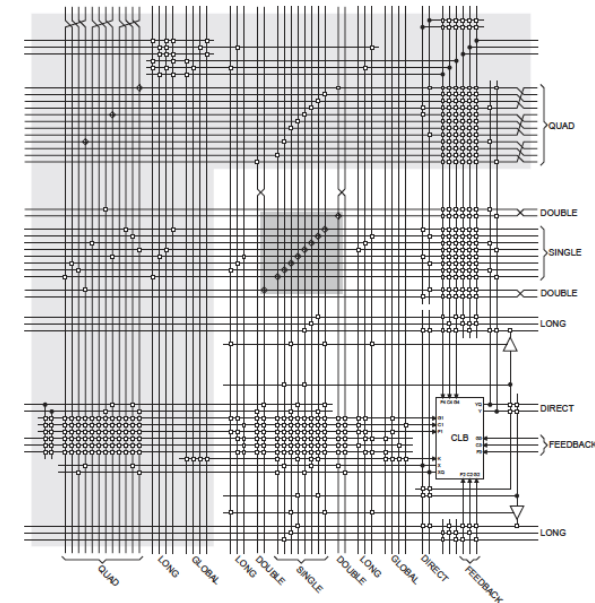
XC4000E and XC4000X Series Features

Note: Information in this data sheet covers the XC4000E, XC4000EX, and XC4000XL families. A separate data sheet covers the XC4000LA and XC4000XV families. Electrical Specifications and package/pin information are covered in separate sections for each family to make the information easier to access, review, and print. For access to these sections, see the Xilinx web site at http://www.xilinx.com/xlnx/web/xil_publications_index.jsp

- System featured Field-Programmable Gate Arrays
 - SelectRAM™ memory, on-chip ultra-fast RAM with synchronous write option
 - dual-port RAM option
 - Fully PCI compliant (speed grades -2 and faster)
 - Abundant flip-flops
 - Flexible function generators
 - Dedicated high-speed carry logic
 - Wide edge decoders on each edge
 - Hierarchy of interconnect lines
 - Internal 3-state bus capability
 - Eight global low-skew clock or signal distribution networks
- System Performance beyond 80 MHz
- Flexible Array Architecture
- Low Power Segmented Routing Architecture
- Systems-Oriented Features
 - IEEE 1149.1-compatible boundary scan logic support
 - Individually programmable output slew rate
 - Programmable input pull-up or pull-down resistors
 - 12 mA sink current per XC4000E output
- Configured by Loading Binary File
 - Unlimited re-programmability
- Read Back Capability
 - Program verification
 - Internal node observability
- Backward Compatible with XC4000 Devices
- Development System runs on most common computer platforms
 - Interfaces to popular design environments
 - Fully automatic mapping, placement and routing
 - Interactive design editor for design optimization

May 14, 1999 (Version 1.6)

Product Obsolete or Under Obsolescence XC4000E and XC4000X Series Field Programmable Gate Arrays



- Common to XC4000E and XC4000X
- XC4000X only
- Programmable Switch Matrix

Figure 27: Detail of Programmable Interconnect Associated with XC4000 Series CLB

6-30

May 14, 1999 (Version 1.6)

Petitioner Microsoft Corporation - Ex. 1035, p. 30

“pass computed data seamlessly between said computational loops”

- Petitioner and Dr. Stone concede in reply that the only basis for claiming Splash2 discloses looping is relying on the infinite “loop-endloop” in Figs. 8.7 and 8.12 to allegedly compare each of the genetic sequences of datum.

2. Splash2 Discloses The Claimed Computational Loops

DirectStream next repeats its argument, rejected in the Institution Decision, that Splash2 does not disclose computational loops, and it makes a similar argument with respect to other prior art cited in the Petition. Response, 91-93, 95. Once again, however, DirectStream bases its arguments solely on its legally improper interpretation, *see id.*, so they should be rejected.¹²

Further, DirectStream’s characterization of the computational loops disclosed in Splash2 simply ignores what the reference says. EX1076¶¶15-19.

Figures 8.7 and 8.12 expressly label the listed instructions, respectively as a “loop,” EX1007, 101, 105, and the text of Splash2 explains that those instructions are executed repeatedly as source and target sequences are shifted through the two systolic arrays, EX1007, 102-04; *see also* EX2064, 225:9-17.

“pass computed data seamlessly between said computational loops”

- Petitioner does not explain how these sequences arrive in Splash2, even though Petitioner and its expert acknowledge the sequences must be “streamed through the array.”

Chapter 8 of Splash2 further discloses the interconnection of the Processing Elements instantiated to carry out the Edit Distance Algorithm. For example, in the Bidirectional Systolic Array source and target sequences are streamed through each Processing Element, as shown in Figure 8.5. EX1007, 101; EX1003¶227. Similarly, in the Unidirectional Systolic Array target sequences are streamed through each Processing Element, as shown in Figure 8.9. EX1007, 103;

- Petition, 33
- EX1003¶134 (“More specifically, for this implementation... two genetic sequences are shifted in opposite directions through multiple processing elements of the Splash 2 system.... The source and target sequences enter the array on opposite ends...”).

“pass computed data seamlessly between said computational loops”

- Petitioner does not dispute that Splash2 requires a host Sparc computer. Petition, 30.
- This workstation controls the sequences of data sent into Splash2 is also consistent with the disclosures in Halverson. See EX2167 at 14-15; EX2164 ¶¶ 42-43.

“pass computed data seamlessly between said computational loops”

- Gokhale is unavailing.
- Despite Dr. Stone professing without any support or analysis that Splash2 is not a SIMD structure, Splash2 clearly states it does operate as a SIMD structure, which stands for single instruction, multiple data. EX1007, 125 (“The Splash 2 system supports several models of computation, including PEs executing a single instruction on multiple data (SIMD mode) and PEs executing multiple instructions on multiple data (MIMD mode).”)

Disputed Claim Terms

- pass computed data “seamlessly” between said computational loops
- “systolic” and “data driven”
- “computational loop”
- “stream communication”

“systolic” and “data driven”

2. Systolic and Data Driven

DirectStream’s Construction

This term has its plain and ordinary meaning and need not be construed.

In the alternative, this term may be construed as:

An array of many interconnected functional units that operates in a data flow sense and allows different data to flow in different directions

Petitioner’s Construction

The characteristic of rhythmically computing and passing data directly between processing elements “without a program counter or clock that drives the movement of data” and operating in a manner that is “transport triggered, *i.e.*, by the arrival of a data object”

“systolic” and “data driven”

- Petitioner’s construction of this term improperly introduces the limitation of “passing data directly” that is not supportable by the intrinsic or extrinsic evidence, and the Board’s institution decision adopted this incorrect construction.
- Similar flaw to Petitioner’s construction for “seamless”
- In contrast, Petitioner does not insert “directly” into its construction for data driven
 - Petition, p. 11 (“... the ordinary meaning to a Skilled Artisan of “data driven” is the scheduling of operations upon the availability of their operands”).

“systolic” and “data driven”

- Petitioner and its expert conflate the concepts of systolic and seamless, removing any functional difference between the two terms.

EX2064 at 85:3-22:

EX2064, 93:3-94:22:

Page 85

1 STONE, Ph.D.
2 Oh, there it is. Okay.
3 So this concept of systolic, what's
4 your opinion of what that means?
5 A. My opinion of what it means is in
6 the report. It means it's the characteristic
7 of rhythmically computing and passing data
8 directly between processing elements. And then
9 I quote: "Without a program counter or clock
10 that drives a movement of data," and also
11 operating in a manner that is, "transport
12 triggered, i.e., by the arrival of a data
13 object."
14 Q. Okay. And you mention the word
15 "directly," it was passing data directly
16 between processing elements. What does that
17 phrase mean to you or what's the context?
18 What are you trying to describe
19 there?
20 A. That the data goes from first to the
21 second without going to something intervening.
22 It directly go -- is connected immediately.
23 Indirectly we -- you go through one or more
24 intervening places to get there.
25 Q. Okay. So would memory, if the data

Page 93

1 STONE, Ph.D.
2 question. Sorry.
3 Q. Sure, yes. So I'm wondering about
4 the inclusion of the word "directly" in your
5 definition for seamlessly like it was in
6 systolic, and I'm wondering, is it your opinion
7 that they're related concepts?
8 A. That seamlessly and systolically are
9 related?
10 Q. Yes.
11 A. In my declaration, I use the word
12 "directly" to construe systolic. I use the
13 word "directly" to construe the word
14 "seamlessly" or the whole phrase contains
15 "seamlessly." So the relation is through the
16 word "directly."
17 Q. Well, tell me if I'm
18 misunderstanding then. So because of the
19 inclusion of the word "directly" in both, it is
20 your opinion that they are related?
21 A. This is -- this is -- in this
22 context, this is how they're related. If you
23 went to different contexts and you said
24 "seamlessly" and you said something else about
25 "systolic," perhaps in that context they

Page 94

1 STONE, Ph.D.
2 wouldn't be related, but in the terms of
3 "directly" here, they are related.
4 Q. Okay. And apologies, we may have to
5 flip back and forth between these two.
6 So in systolic, your definition
7 included: "Passing data directly between
8 processing elements."
9 Do you recall that?
10 A. I do.
11 Q. And then for "seamlessly," the
12 phrase is: "Communicate computed data directly
13 between functional units."
14 Do you see that?
15 A. I see that.
16 Q. Okay. Is it the same data that
17 you're referring to in those two contexts?
18 A. It -- there may be some relation,
19 but the -- specifically the limitation that I'm
20 referring to for seamlessly talks about
21 computed data, output data. Systolically is
22 not limited to output data or computed data.
23 Q. Okay. Then do you see there's -- I
24 guess what's being communicated to -- or sorry.
25 Let me try that again.

“systolic” and “data driven”

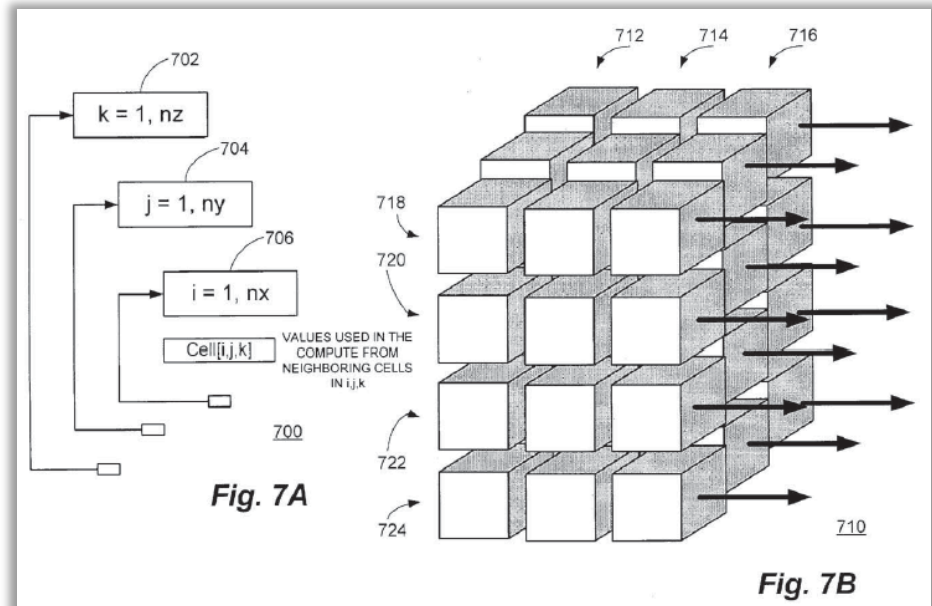
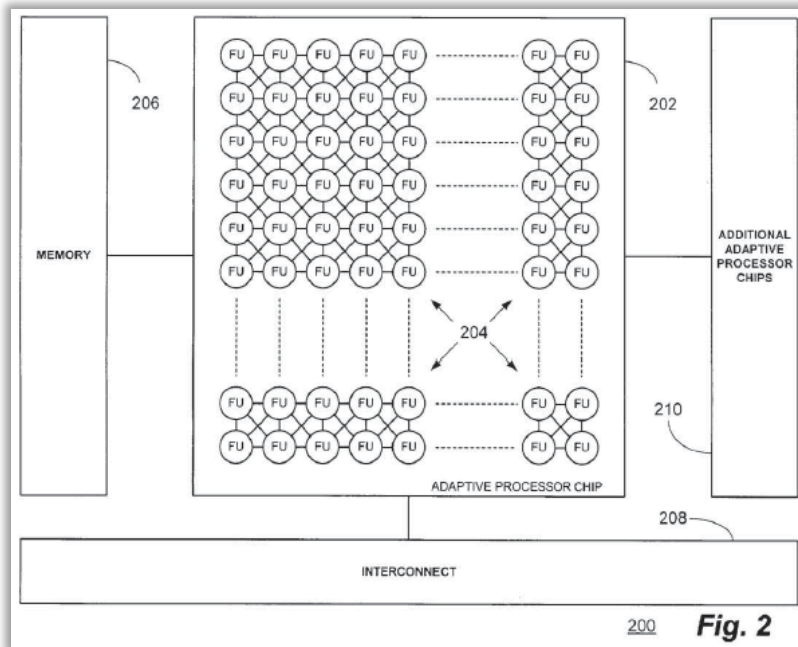
- "Dr. Kung described a systolic system as follows:
 - A systolic system consists of a set of interconnected cells, each capable of performing some simple operation. Because simple, regular communication and control structure have substantial advantages over complicated ones in design and implementation, cells in a systolic system are typically interconnected to form a systolic array or a systolic tree. Information in a systolic system flows between cells in a pipelined fashion, and communication with the outside world occurs only at the “boundary cells.” For example, in a systolic array, only those cells on the array boundaries may be I/O ports for the system. ... The basic principle of a systolic architecture, a systolic array in particular, is illustrated in Figure 1. By replacing a single processing element with an array of PEs[processing elements], or cells in the terminology of this article, a higher computation throughput can be achieved without increasing memory bandwidth. The function of the memory in the diagram is analogous to that of the heart; it “pulses” data (instead of blood) through the array of cells. The crux of this approach is to ensure that once a data item is brought out from the memory it can be used effectively at each cell it passes while being “pumped” from cell to cell along the array. This is possible for a wide class of compute-bound computations where multiple operations are performed on each data item in a repetitive manner.
- EX1016 at 39."

“systolic” and “data driven”

- Systolic means an array of interconnected processing elements that only interact with memory at the array boundaries so that the data is processed by multiple processing elements before returning to memory.
 - See also EX2040 at 1 (“The term systolic arrays was coined by Kung ... to describe application specific VLSI architectures that were regular, locally connected and massively parallel with simple processing elements (PEs).”).
- Memory acts like the heart in systolic system by “pulsing” data into the array where it is pumped from processing element to processing element before returning to memory. EX2046¶16.

“systolic” and “data driven”

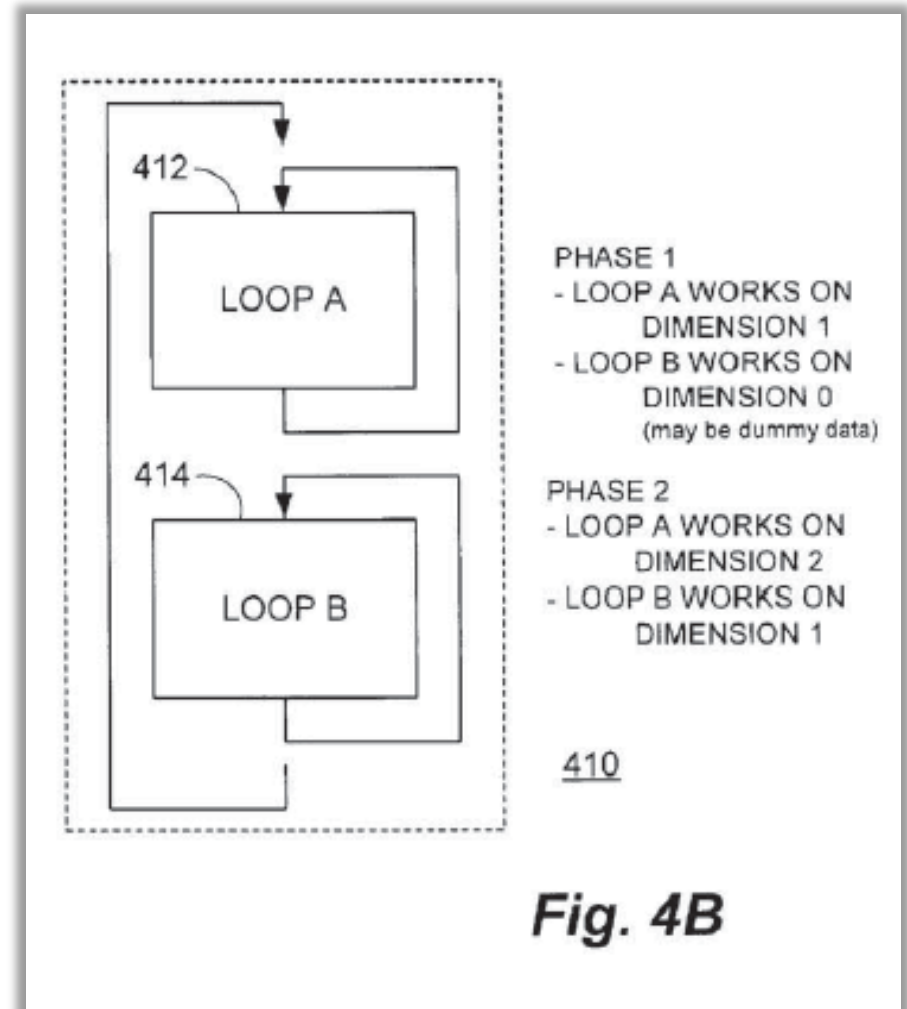
- Figure 2 shows a systolic array of interconnected function units (which as explained above are each “a set of logic that performs a specific operation”) that interact only with memory at the boundaries of the array:



- EX1001, Figs. 2, 7B

“systolic” and “data driven”

- The '324 patent also talks about how this improves performance because the “boundary data that is shared between these functional units need never leave a single integrated circuit chip.” EX1001, 2:38-42; EX2111 ¶¶125-131.
- And in the '324 patent, the “[s]ystolic implementation will connect computational loops such that data from one loop will be passed as input data to a concurrently executing compute loop.” EX1002 at 226.



“systolic” and “data driven”

- EX1001 at 6:21-30.
 - In contrast to the sequential processing operation 400 (FIG. 4A) the solution to the problem of most effectively utilizing available resources is to have an application evaluate a problem in a data flow sense. That is, it will “pass” a subsequent dimension of a given problem through the first loop 412 of logic concurrently with the previous dimension of data being processed through the second loop 414. In practice, a “dimension” of data can be: multiple vectors of a problem, multiple planes of a problem, multiple time steps in a problem and so forth.

“systolic” and “data driven”

- During prosecution, the applicant confirmed that it was using the plain meaning of “systolic”:
 - Instantiation is a term well known to one of ordinary skill in the art of reconfigurable processing... Similarly the term systolic computation is derived from continual and pulsating pumping of the human heart. In computer architecture a systolic array is an arrangement of data processing units similar to a central processing unit but without a program counter or clock that drives the movement of data. That is because the operation of the systolic array is transport triggered, i.e. by the arrival of a data object. Data flows across the array between functional units, usually with different data flowing in different directions. David J. Evans in his work, *Systolic algorithms...* define[s] a Systolic system as a “network of processors which rhythmically compute an[d] pass data through the system.”
- EX1002 at 225-26

“systolic” and “data driven”

- In contrast, Petitioner’s construction ignores this plain and customary understanding of the term “systolic” within the context of the ’324 Patent.
- Just as with Petitioner’s error with respect to “seamless,” Petitioner once again improperly inserts the limitation of “directly” without any support from the intrinsic or extrinsic evidence.
- However, this is nonsensical because it would exclude standard FPGAs (including the type described in the embodiments of the ’324 Patent and the specific FPGA chips used in Petitioner’s prior art references) since standard FPGAs contain reconfigurable routing resources (comprising buffers and switches) between the configurable logic blocks. For example, the literature on Xilinx FPGA chips shows buffer switch boxes and three state buffers to connect two or more configurable logic blocks. EX1035 at 31; EX2078 at 19-29, 32-34, 37-41, 46-51, 59-65.

“systolic” and “data driven”

- Petitioner’s construction is based on arguing that SRC acted as its own lexicographer. 601 Petition at 14-15.
- But the standard for “finding lexicography” is exacting and requires the patentee to “clearly set forth a definition of the disputed claim term” and “clearly express an intent to define the term.”
 - *Pacing Techs., LLC v. Garmin Int’l, Inc.*, 778 F.3d 1021, 1024 (Fed. Cir. 2015) (“Disavowal, or disclaimer of claim scope, is only considered when it is clear and unmistakable.”).
 - *Ancora Techs., Inc. v. Apple, Inc.*, 744 F.3d 732, 734 (Fed. Cir. 2014) (“A claim term should be given its ordinary meaning in the pertinent context, unless the patentee has made clear its adoption of a different definition or otherwise disclaimed that meaning.”).

“systolic” and “data driven”

- The only time the applicant describes what it meant when it used the term “systolic” comes from the following passage on the next page:
 - Thus in Applicant’s invention Systolic implementation will connect computational loops such that data from one compute loop will be passed as input data to a concurrently executing compute loop. In the Applicant’s invention data computed by computation units or groups of functional units flows seamlessly and concurrently with data being computed by other groups of functional units. EX1002 at 226.
- If anything were to be construed as an explicit definition of the term “systolic” it should be that sentence. But this description is in accordance with the term’s plain and customary meaning.

“systolic” and “data driven”

- The limitation “passing data directly between processing elements is simply not a requirement of systolic systems. EX2111¶177.
- And the limitations of (i) “without a program counter or clock that drives the movement of data” and (ii) “operating in a manner that is ‘transport triggered, i.e., by the arrival of a data object’” are redundant at best.
- Transport triggered operations do not utilize program counters or a clock to drive the movement of data because they are triggered by the availability of inputs. EX2046¶¶14, 16; EX2047 at 1 (“[I]n data-driven (e.g., data-flow) computers the availability of operands triggers the execution of the operation to be performed on them...”).

“systolic” and “data driven”

- This contrasts with traditional Von Neumann computers that must use program counters and/or clocks to drive data movement because of their sequential, centralized control scheme. EX2046 ¶¶9-10;
- EX2048 at 2 (“In a data flow computer, an instruction is ready for execution when its operands have arrived. There is no concept of control flow, and data flow computers do not have program location counters.”) (emphasis added).
- So even if the Board were inclined to construe this term, there is no reason to require both “transport triggered” and “without a program counter or clock that drives the movement of data.”

Disputed Claim Terms

- pass computed data “seamlessly” between said computational loops
- “systolic” and “data driven”
- **“computational loop”**
- “stream communication”

“computational loop”

C. “computational loop” ... [wherein only functional units needed to solve the calculation are instantiated]

DirectStream’s Construction	Petitioner’s Construction
a set of computations that is executed repeatedly per datum, either a fixed	Petitioner has not proposed any construction for this term
number of times or until some condition is true or false	

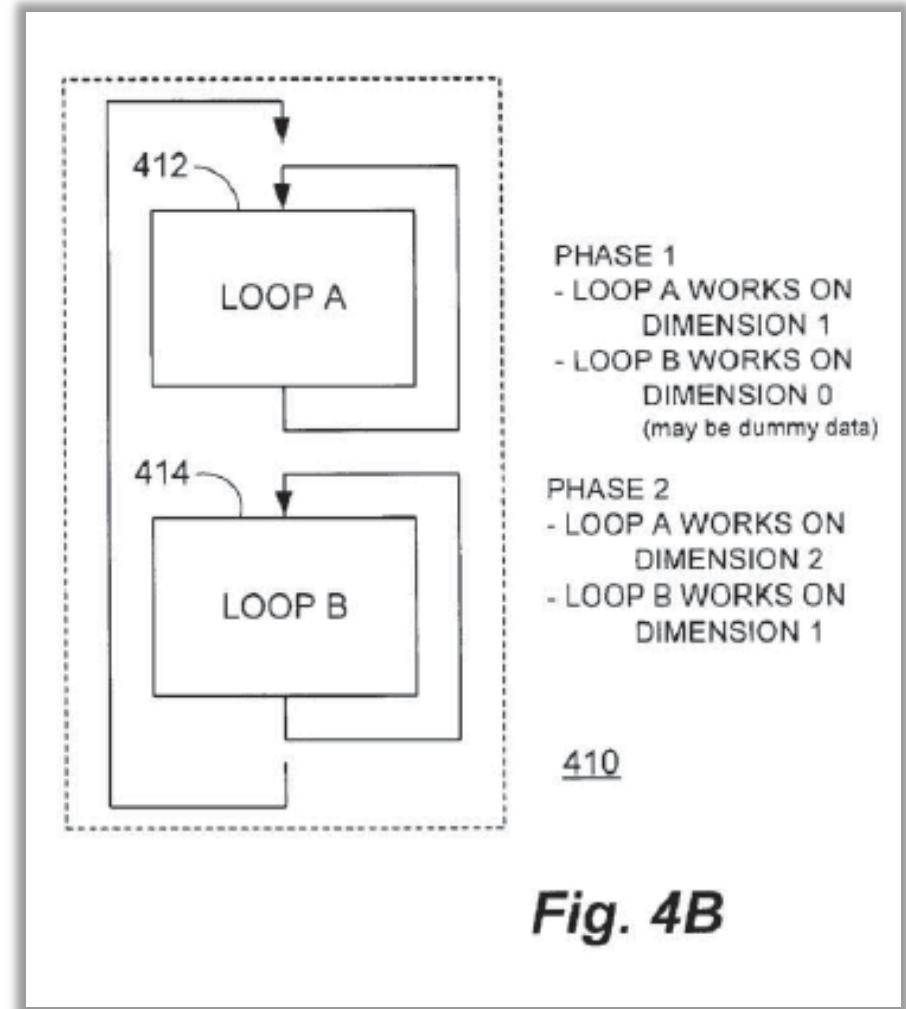
- At Institution, the Board construed the term to mean “a set of computations that is executed repeatedly, either a fixed number of times or until some condition is true or false.” Paper 21.

“computational loop”

- ... the word “computational” simply means an act, process, or method of computing. EX2038 at 3 (definition of “computation”). However, the plain language of the independent claims and the specification further clarify that the computations are part of the calculations for which the functional units are being instantiated. See EX2111 ¶¶125-131.
- Loop: ... A set of statements in a program executed repeatedly, either a fixed number of times or until some condition is true or false. EX 2026 at 8 (Microsoft Press Computer Dictionary Third Edition 1997).
- Loop: in a computer, a series of instruction being carried out repeatedly until a terminal condition prevails. EX2025 at 5 (Modern Dictionary of Electronics Sixth Edition 1997).
- Loop: a sequence of instructions that is repeated until a prescribed condition, such as agreement with a data element or completion of a count, is satisfied. EX2024 at 4 (Oxford Dictionary of Computing Fourth Edition 1997).

“computational loop”

- This definition is consistent with how the '324 patent's specification utilizes the term “loop.” For example, the specification depicts numerous “loops” that are repeated until some condition is met.
- EX1001 at Fig. 4A, 4B



“computational loop”

- DirectStream’s definition is also consistent with how “loop” is used in “Delivering Acceleration: The Potential for Increased HPC Application Performance Using Reconfigurable Logic, which was incorporated by reference into the ’324 patent. EX1001 at 4:59-63; EX2037 at 4, 5, 7, 12, 13, 16, 17, 18, 19.
 - Discussing how SRC converted algorithms written in a high-level language (such as FORTRAN or C) by using a compiler to generate a “data flow graph” that was further optimized manually into “an algorithm data flow that will be put into hardware logic for the FPGAs.” EX2037 at 7, 10-11.

“computational loop”

- Additionally, the '666 patent further illustrates the concept of a computational loop as would be known to a POSITA in the context of describing the use of stream communications:
 - Consumer loops are simple iterative processes that operate to provide a particular result. As a simple example, an addition operation may necessitate an iterative loop until a certain value is obtained. Consumer loops receive or fetch data values from a buffer and begin the looping process. Once launched, the computations continue until completed. Thus, using our simple addition example, the loop computation may comprise fetching a value, adding the value to the existing total, and then comparing the value to a predetermined number to determine if a termination criteria has been reached. This process may take two or three clock ticks of the processor. Thus even though there is additional data in the buffer available to the consumer, there is a lag between when a value has been fetched and when the loop has determined that it should terminate.
- EX2027 at 2:64-3:23, 6:6-28.

“computational loop”

EX2027 at 2:64-3:23



US08589666B2

(12) **United States Patent**
Hammes (10) **Patent No.:** **US 8,589,666 B2**
 (45) **Date of Patent:** **Nov. 19, 2013**

(54) **ELIMINATION OF STREAM CONSUMER LOOP OVERSHOOT EFFECTS** (56) **References Cited**
 U.S. PATENT DOCUMENTS

(75) **Inventor:** Jeffrey Hammes, Colorado Springs, CO (US)
 7,085,955 B2* 8/2006 Prabhu 714.6
 2002/0124159 A1* 9/2002 Bekooji et al. 712/226

(73) **Assignee:** SRC Computers, Inc., Colorado Springs, CO (US)
 * cited by examiner

(*) **Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1985 days.
Primary Examiner — Andrew Caldwell
Assistant Examiner — George Giroux
 (74) *Attorney, Agent, or Firm* — William J. Kubicki; Hogan Lovells US LLP

(21) **Appl. No.:** 11/456,466 (57) **ABSTRACT**
 A reconfigurable processor invoking data stream pipelining is configured to associate a restore buffer with each incoming data stream. The buffer is configured to be of sufficient size to maintain data values dispatched to a loop so as to restore values fetched and lost due to loop overshoots. The restore buffer stores the values that were recently fetched from the buffer to the loop. To determine how many data values should be restored, the loop counts the number of the data values it takes from each data stream and the number of valid loop iterations that take place. Once a loop termination is detected, the loop halts the fetching of values from the restore buffer and compares, for each stream, the number of loop iterations with the number of values fetched. The difference is the number of extra values that were taken from the restore buffer and are restored.

(22) **Filed:** Jul. 10, 2006

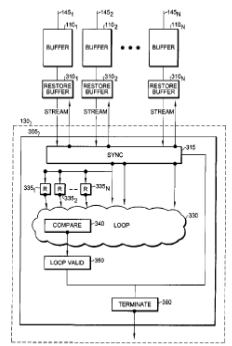
(65) **Prior Publication Data**
 US 2008/0010444 A1 Jan. 10, 2008

(51) **Int. Cl.**
G06F 15/00 (2006.01)
G06F 9/38 (2006.01)
G06F 9/00 (2006.01)
G06F 9/44 (2006.01)

(52) **U.S. Cl.**
 USPC 712/241; 714/18

(58) **Field of Classification Search**
 USPC 712/241, 201, 25
 See application file for complete search history.

22 Claims, 6 Drawing Sheets



Patent Owner Saint Regis Mohawk Tribe
 Ex. 2027 - p. 1

interconnection of functional units, and that are used to create specialized logic configurations. The dataflow graphs are analyzed and optimized to provide an optimal set of logic configurations for each particular high level program. For example, a data buffer in a loop structure can be designed based on the program computations that are going to be run

with sometimes be empty and the consumer will wait for new values to be available.

When loops, in a high level language such as C or Fortran, are compiled for execution on reconfigurable hardware such as FPGAs, aggressive pipelining is employed, with the goal of achieving as great a throughput of data as possible. The

Patent Owner Saint Regis Mohawk Tribe
 Ex. 2027 - p. 8

US 8,589,666 B2

3

body of an inner loop is converted to a dataflow graph with many individual functional units that are derived from the operators in the loop body. The FPGA implementations of these functional units may have latencies of one or more clock ticks, and the interconnection of many functional units will result in a loop body that can have new data fed in on every clock tick, even though it may take many clock ticks for the results of an iteration's data to reach the bottom of the dataflow graph. Thus many iterations are active at any point in time.

Part of the loop body's dataflow graph consists of the expression that computes a loop termination criteria. If a new loop iteration is activated on every clock tick, then extra loop iterations are activated despite a termination value being achieved. For example, when it takes three clocks for the data of a given loop iteration to produce the true/false termination value, then by the time the termination expression becomes true, three additional iterations of the loop will already have started. These extra iterations are harmless as long as they are not allowed to affect the state of the overall computation. However, the data acquired by the loop from the data stream may be lost.

Consider the following producer-consumer loops:

4

and the Loop Valid node 250 latches into a false state, telling the Terminate node 260 to emit a "done" pulse telling the Synchronization node 215 that the loop has reached its termination condition.

Because of pipelining, it can take multiple clock ticks for the effects of a value to move through the loop body 230, but when there are values available in the FIFO buffer 110, the Synchronization node 215 will take a value from the FIFO buffer and start a new loop iteration 235 on every clock tick. Assuming that the Compare node 240 and Loop Valid node 250 each have a one-clock tick latency, the Synchronization node 215 will spawn at least of couple of extra iterations before it sees that loop termination has occurred, and those extra loop iterations will have taken from the FIFO buffer values that should not have been taken.

This effect becomes even more pronounced when the termination expression is more complex. A rather extreme case arises from the following: for (i=0; i<w/v; i++) The "divide operation" is typically a many-clock latency operation; thirty or more clock ticks is not unusual, so the Synchronization node 215 of this loop 230 could spawn more than thirty extra iterations before it learns that the loop has reached termination.

The effect of these extra iterations on the consumer loop nest is twofold. First, when the inner loop is entered for the

“computational loop”

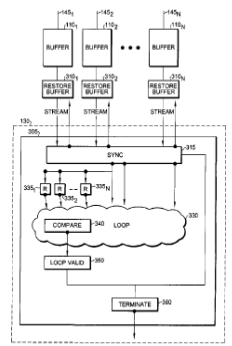
EX2027 at 6:6-28.



US008589666B2

(12) **United States Patent**
Hammes (10) **Patent No.:** **US 8,589,666 B2**
(45) **Date of Patent:** **Nov. 19, 2013**

(54) **ELIMINATION OF STREAM CONSUMER LOOP OVERSHOOT EFFECTS** (56) **References Cited**
U.S. PATENT DOCUMENTS
(75) **Inventor:** **Jeffrey Hammes**, Colorado Springs, CO (US) 7,085,955 B2* 8/2006 Prabhu 7146
2002/0124159 A1* 9/2002 Bedooji et al. 712/226
(73) **Assignee:** **SRC Computers, Inc.**, Colorado Springs, CO (US) * cited by examiner
Primary Examiner — Andrew Caldwell
Assistant Examiner — George Giroux
(*) **Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1985 days.
(74) **Attorney, Agent, or Firm:** William J. Kubicki; Hogan Lovells US LLP
(21) **Appl. No.:** **11/456,466** (57) **ABSTRACT**
A reconfigurable processor invoking data stream pipelining is configured to associate a restore buffer with each incoming data stream. The buffer is configured to be of sufficient size to maintain data values dispatched to a loop so as to restore values fetched and lost due to loop overshoots. The restore buffer stores the values that were recently fetched from the buffer to the loop. To determine how many data values should be restored, the loop counts the number of the data values it takes from each data stream and the number of valid loop iterations that take place. Once a loop termination is detected, the loop halts the fetching of values from the restore buffer and compares, for each stream, the number of loop iterations with the number of values fetched. The difference is the number of extra values that were taken from the restore buffer and are restored.
(22) **Filed:** **Jul. 10, 2006**
(65) **Prior Publication Data**
US 2008/0010444 A1 Jan. 10, 2008
(51) **Int. Cl.**
G06F 15/00 (2006.01)
G06F 7/38 (2006.01)
G06F 9/00 (2006.01)
G06F 9/44 (2006.01)
(52) **U.S. Cl.** **712/241; 714/18**
USPC
(58) **Field of Classification Search** **USPC** **712/241; 201; 25**
See application file for complete search history.
22 Claims, 6 Drawing Sheets



Patent Owner Saint Regis Mohawk Trib...
Ex. 2027 - p. 1

US 8,589,666 B2

5 6

invokes data stream pipelining during iterative computations, an FPGA or other reconfigured processor is configured to place two buffers between the producer of a data stream and the data consumer. In addition to a typically found FIFO buffer, a restore buffer is interposed between the producer and the consumer of the data values. The restore buffer is configured to be of sufficient size to maintain a persistent memory of enough data values dispatched to the iterative loop structure so as to restore data values fetched and lost due to pipelined consumer loop overshoots.

In one aspect of the present invention, a restore buffer is placed between each incoming stream and its consumer loop. The restore buffer stores the data values that were recently fetched from the buffer to the consumer so that later they can be restored. To determine how many data values should be restored, the iterative loop structure counts the number of the data values it takes from each data stream. The iterative loop structure also counts the number of valid loop iterations that take place. Once a loop termination is detected, the iterative loop structure halts fetching data values from the restore buffer and compares, for each stream, the number of loop iterations with the number of values fetched from each data stream. The difference is the number of extra data values that were taken from the restore buffer and must be restored. The iterative loop structure then directs the restore buffers to restore those values.

The foregoing and other features, utilities and advantages of the invention will be apparent from the following more particular description of an embodiment of the invention as illustrated in the accompanying drawings.

tion of pipelined data streams are discussed in detail herein. In a pipelined data stream environment comprising a producer of data and a consumer of that data, the effects of consumer loop overshoots are eliminated by interposing, in one embodiment of the present invention, two buffers between the producer and consumer. As previously described, a typical configuration of a pipelined data stream structure places a single buffer between the producer and the consumer. Such a single buffer is typically configured to absorb differences in the production and consumption rate of the data streams, but does little to prevent the lost of data due to consumer loop overshoots.

Consumer loops are simple iterative processes that operate to provide a particular result. As a simple example, an addition operation may necessitate an iterative loop until a certain value is obtained. Consumer loops receive or fetch data values from a buffer and begin the looping process. Once launched, the computations continue until completed. Thus, using our simple addition example, the loop computation may comprise fetching a value, adding the value to the existing total, and then comparing the value to a predetermined number to determine if a termination criteria has been reached. This process may take two or three clock ticks of the processor. Thus even though there is additional data in the buffer available to the consumer, there is a lag between when a value has been fetched and when the loop has determined that it should terminate.

“computational loop”

- In contrast, Petitioner and its expert do not even provide a construction for this term. Instead, Petitioner’s expert merely assumes that a computational loop is present because multiple data are being processed. EX2064 at 178:17-180:11.

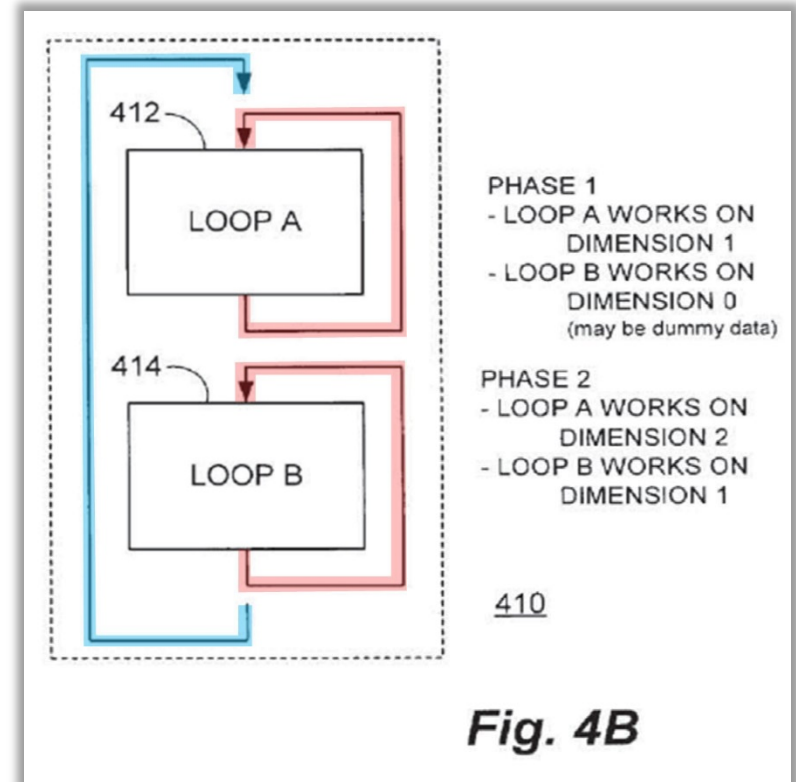
<p style="text-align: right;">Page 178</p> <p>1 STONE, Ph.D. 2 Q. Okay. Okay. I appreciate the -- 3 you walking through that. That is helpful. 4 So looking at this bidirectional -- 5 actually it may be better to look at the 6 Figure 8.7 that contain the -- the code. I 7 think in your report, it was on Page 121. 8 A. So I should go back to the report, 9 but I'll keep this handy. 10 Q. Okay. Yeah. It should be the same 11 excerpt, so it should be Figure 8.7. 12 A. Okay. And the page number again? 13 Q. Page 121. 14 A. Okay. I'm there. 15 Q. Do you see that figure? 16 A. I do. 17 Q. Is that a computational loop, 18 Figure 8.7? 19 A. Well, it is, and I'm informed 20 because it says "loop" and "end loop." This is 21 a loop. 22 Q. Well, in terms of the claims, 23 though, referencing a computational loop, do 24 you recall that -- that claim term? 25 A. Yes.</p>	<p style="text-align: right;">Page 179</p> <p>1 STONE, Ph.D. 2 Q. Is it your opinion that that loop, 3 from the loop to the end loop, that code is a 4 computational loop? 5 A. Everything between loop and end 6 loop, including the words "loop" and "end 7 loop," comprise a computational loop. 8 Q. And so the claims in, I believe, 9 both the '324 and the '800 patent refer to a 10 first and second computational loop. 11 Do you recall that? 12 A. I do. 13 Q. So is it your opinion that both of 14 those computational loops are the same loop 15 code that's in Figure 8.7? 16 MR. MICALLEF: Objection. Form. 17 THE WITNESS: I have already 18 testified that when you build these cells, 19 you don't build them the way the software 20 would read that. You would use VHDL and 21 convert this into parallel implementation. 22 So the cells that would be in the Splash 2 23 array would be the VHDL instantiation of 24 this loop, and each of those instantiations 25 would be the same loop, but they're</p>
<p style="text-align: right;">Page 180</p> <p>1 STONE, Ph.D. 2 distinct, distinct hardware. 3 BY MR. HSU: 4 Q. Right. And so let me ask it this 5 way: So your opinion for this bidirectional 6 array, both of those distinct instantiations 7 would be performing the same loop software code 8 separately instantiated, but same instruction? 9 A. After the instantiation, the 10 hardware would be the same. The hardware would 11 be the same. 12 Q. Right, right. Okay. 13 MR. MICALLEF: Counsel, how long 14 have we been going? 15 MR. HSU: Oh, yeah. Let's take a 16 break. I think it's been about an hour 17 since our last one. 18 THE VIDEOGRAPHER: We're going off 19 the record. The time is 3:18 p.m. 20 (Recess 3:18-3:29.) 21 THE VIDEOGRAPHER: We're going back 22 on the record. The time is 3:29 p.m. 23 EXAMINATION (Continuing) 24 BY MR. HSU: 25 Q. So let me introduce another exhibit</p>	<p style="text-align: right;">Page 181</p> <p>1 STONE, Ph.D. 2 really quick. This one should be Exhibit 1011. 3 Did it come up? Yes. 4 So do you recognize Exhibit 1011? 5 A. Yes. 6 Q. And is it okay if we refer to this 7 one as Chunky SLD? 8 A. Sure. 9 Q. I believe that's the shorthand you 10 used in your declaration, so we'll try to keep 11 it consistent. 12 A. Yes. 13 Q. So is it correct that Chunky SLD is 14 being implemented on a Splash 2 system? 15 A. Yes. 16 Q. And can you describe what the 17 algorithm is that they're trying to perform in 18 Chunky SLD? 19 A. They're looking for a pattern in an 20 SAR -- that's "synthetic aperture radar" 21 image -- and they're trying to find out where 22 and if it might exist within that image. 23 Q. Okay. And if you could turn with me 24 to actually hang on. I think this is Page 194. 25 Do you see there's a ShapeSum</p>

“computational loop”

- Petitioner incorrectly claims “DirectStream’s expert, Dr. Homayoun, never offers an interpretation of ‘computational loop’... [and] DirectStream could not convince him to support its position.” Reply, 36-37.
- EX2111, ¶207 (Dr. Homayoun opining that “A computational loop evaluates each piece of data multiple times, ‘a fixed number of times or until some condition is true or false,’” and more importantly, opining throughout this section how Petitioner’s flawed view of the claim term results in a flawed invalidity analysis).

“computational loop”

- Specifically, Fig.4B of the patent (which patentee used to distinguish from the prior art depicted in Fig.4A) depicts two sets of loops: the red loops that form Loop A and Loop B, and the blue program loop that repeats the execution of both in each “phase.” Specifically, the patent states that in “Phase 1,” both loops are active with Loop A working on dimension 1 of the data, and Loop B working on dimension 0 of the data. Then in the next phase (“Phase 2”), both loops are again active with Loop A working on dimension 2 of the data, and Loop B working on dimension 1 of the data.

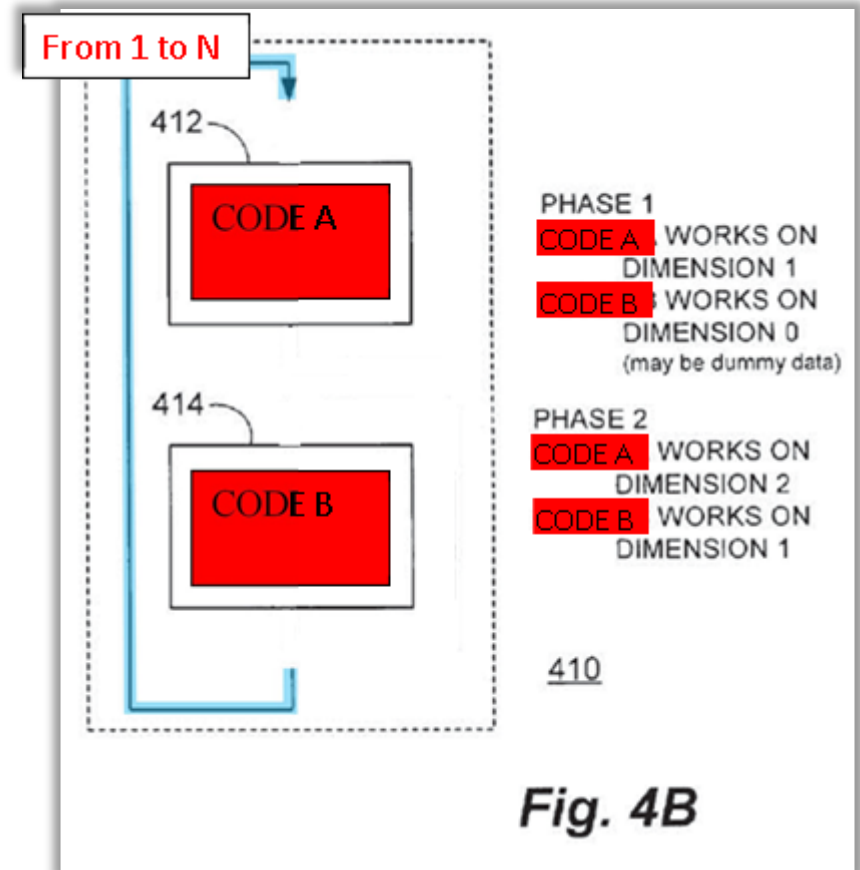


“computational loop”

- Petitioner’s Reply reiterates its flawed interpretation that the red computational loops need only “execute[] instructions on one piece of data, and then execute[] those very same instructions on a next piece of data.” Reply, 35.

“computational loop”

- This argument effectively deletes the red loops as follows:
- Only the blue program loop would be needed to cycle through all of the datum and execute the code once per datum. "

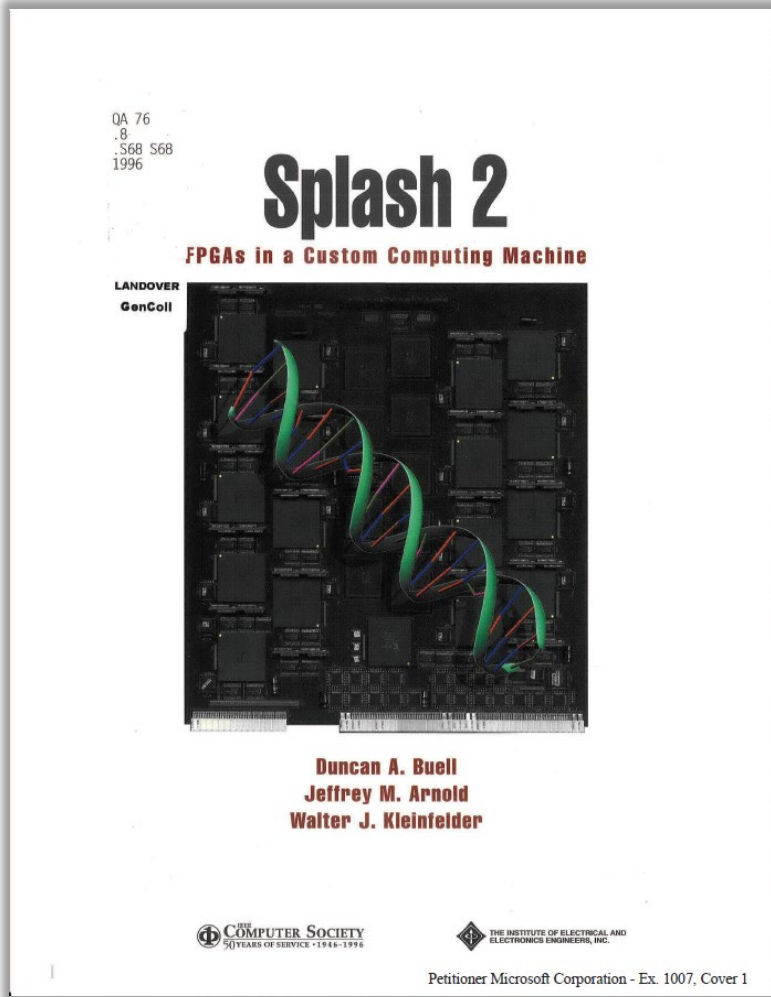


“computational loop”

- Ambiguous at best on disclosure of pseudocode in Splash2 system and how the looping is handled
- Equally plausible interpretation

“computational loop”

- EX1007 Splash2 prior art
- Splash2 does not disclose computational loops as properly construed



“computational loop”

- A “computational loop” is an iterative sequence of computations that repeats until a prescribed condition is satisfied. EX2024 at 4 (definition of “loop”), EX2025 at 5 (same), EX2026 at 8 (same), EX2038 at 3 (definition of “computation”). There is no disclosure of looping or repeating of a computation multiple times for each data until a condition is met or a number of repetitions has been satisfied, as required by the Board’s claim construction and the ’324 Patent. EX2111 ¶¶194-209
- Petitioner’s expert merely assumes that a computational loop is present because multiple data are being processed. EX2064 at 178:17-180:11.

“computational loop”

- Specifically, Fig. 4B from the '324 Patent represents the concept of a nested loop, with a larger loop repeating across the number of data to be processed, and a nested inner loop repeating a number of times for each datum to be processed. EX2111 ¶¶125-130; EX1001 at 3:35-39, 6:1-30, Fig. 4B.
- In other words, the sequence of computations in any of the computational loops is performed on each datum until a prescribed condition is satisfied for that computational loop. EX2111 ¶127; EX1001 at 3:35-39, 6:1-30

“computational loop”

- Then, the computed data is sent from that computational loop, and a second datum is received for the computational loop to run the same sequence of computations until the prescribed condition is once again met for this second datum. EX2111¶¶127; EX1001 at 3:35-39, 6:1-30
- This is not a trivial problem for reconfigurable processors, especially FPGAs. See EX2164¶¶¶39, 43 (discussing the complications with nested looping and inability of Splash2 and other prior art to handle such operations, as was known to a POSITA at the time of the invention).

“computational loop”

- Figures 8.7 and 8.12 each depict an “else if” conditional statement within the framework of a “loop-endloop.” This conditional statement merely selects an execution path for the processor, not a loop that the processor repeats.

```
loop
  if (SCin  $\neq$   $\emptyset$ ) and (TCin  $\neq$   $\emptyset$ ) then
    PEDist  $\leftarrow$  min  $\left\{ \begin{array}{l} \text{PEDist} + \psi(\text{SCin}, \text{TCin}), \\ \text{TDin} + \psi(\text{SCin}, \emptyset), \\ \text{SDin} + \psi(\emptyset, \text{TCin}) \end{array} \right.$ 
  else-if (SCin  $\neq$   $\emptyset$ ) then
    PEDist  $\leftarrow$  SDin
  else-if (TCin  $\neq$   $\emptyset$ ) then
    PEDist  $\leftarrow$  TDin
  endif
  SCout  $\leftarrow$  SCin
  TCout  $\leftarrow$  TCin
  SDout  $\leftarrow$  PEDist
  TDout  $\leftarrow$  PEDist
endloop
```

```
loop
  if (TAGin = SR) then
    if (SRCch =  $\emptyset$ ) then
      SRCch  $\leftarrow$  CHRin
      CHRout  $\leftarrow$   $\emptyset$ 
      DSTout  $\leftarrow$  PDSTin
    else
      CHRout  $\leftarrow$  CHRin
    endif
    PDSTout  $\leftarrow$  PDSTin
  else-if (TAGin = PR) then
    if (SRCch =  $\emptyset$ ) then
      DSTout  $\leftarrow$  PDSTin
    endif
    PDSTout  $\leftarrow$  DSTin
    CHRout  $\leftarrow$  CHRin
  else-if (TAGin = TG) then
    if (SRCch  $\neq$   $\emptyset$ ) and (CHRin  $\neq$   $\emptyset$ ) then
      DSTout  $\leftarrow$  min  $\left\{ \begin{array}{l} \text{PDSTout} + \psi(\text{SRCch}, \text{CHRin}), \\ \text{DSTin} + \psi(\text{SRCch}, \emptyset), \\ \text{DSTout} + \psi(\emptyset, \text{CHRin}) \end{array} \right.$ 
    else-if (SRCch =  $\emptyset$ ) then
      DSTout  $\leftarrow$  DSTin
    endif
    PDSTout  $\leftarrow$  DSTin
    CHRout  $\leftarrow$  CHRin
  endif
  TAGout  $\leftarrow$  TAGin
endloop
```

“computational loop”

- A conditional statement is defined as “a programming-language statement that selects an execution path based on whether some condition is true or false (for example, the IF statement).” EX2026 at 7 (Microsoft Dictionary definition of “conditional statement”); EX2111¶198.

conditional statement \kən-dish'ə-nəl stāt'mənt\ *n.* A programming-language statement that selects an execution path based on whether some condition is true or false (for example, the IF statement). *See also* case statement, conditional, IF statement, statement.

- That is exactly what the code in Figures 8.7 and 8.12 executes through the if-else-if statements. EX2111¶¶198-201.

“computational loop”

- More problematically, the code clearly only runs once per datum. EX2111¶¶204-208. There are no conditions for the “loop-endloop,” which appears to run ad infinitum. EX2111¶¶200-203.
- But this pseudocode in Splash2 cannot be read to run infinitely per datum because the system would then be stuck in an infinite loop on the very first data value.
- The only possible way to interpret this pseudocode in Splash2 (and the only way a POSITA would understand this pseudocode to possibly work) would be to assume that additional code would be created to replace the “loop-endloop” syntax in order to govern the transport of data such that the rest of the pseudocode repeats once for each datum to be transported. EX2111¶¶204-206.

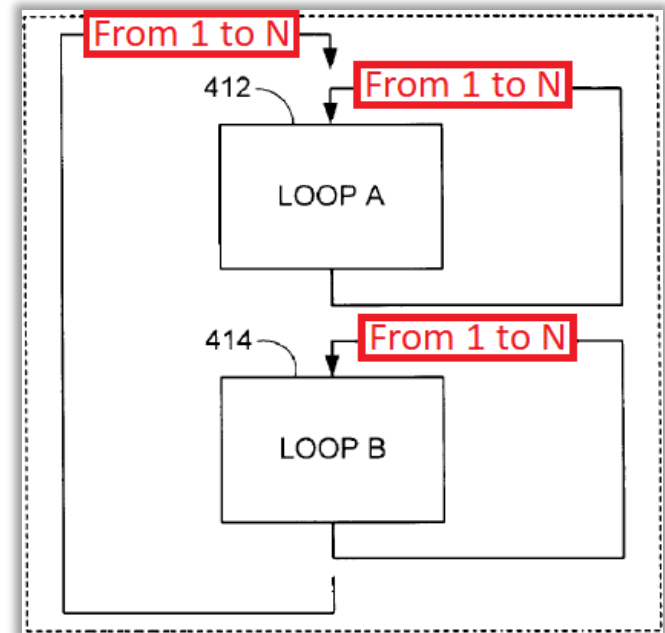
“computational loop”

- A POSITA would recognize this as a program subroutine that is executed once per datum, not a computational loop executing repeatedly per datum until a condition is met. EX2111 ¶¶196-208.”
- This is confirmed by Petitioner and its expert that the “loop-endloop” repeats until the amount of data concludes. 601 Petition at 39-41 (identifying the string target characters to be processed by each processing element); EX 2066 at 225:9- 226:5.

9 Q. Can you explain in what way the
10 statements in Figure 8.7 when implemented would
11 loop?
12 MR. HSU: Objection. Form.
13 THE WITNESS: Well, they would loop
14 because they do one iteration of a loop,
15 and as you pass data through that in
16 subsequent cycles, they do the next
17 iteration, the next iteration.

“computational loop”

- Even assuming arguendo that the amount of data itself could constitute the condition for exiting any of the “loop-endloops,” the pseudocode in Splash2 would not make sense when mapped to Fig. 4B of the '324 Patent:
- Each of the loops cannot be processing from 1 to N, otherwise the Splash2 algorithms would not work as intended, as described even by Petitioner’s own expert. EX2111 ¶¶203-204; EX2064 at 147:6-154:23 (describing intended operation where each datum proceeds through the if-else-if statement of a processing element only once).

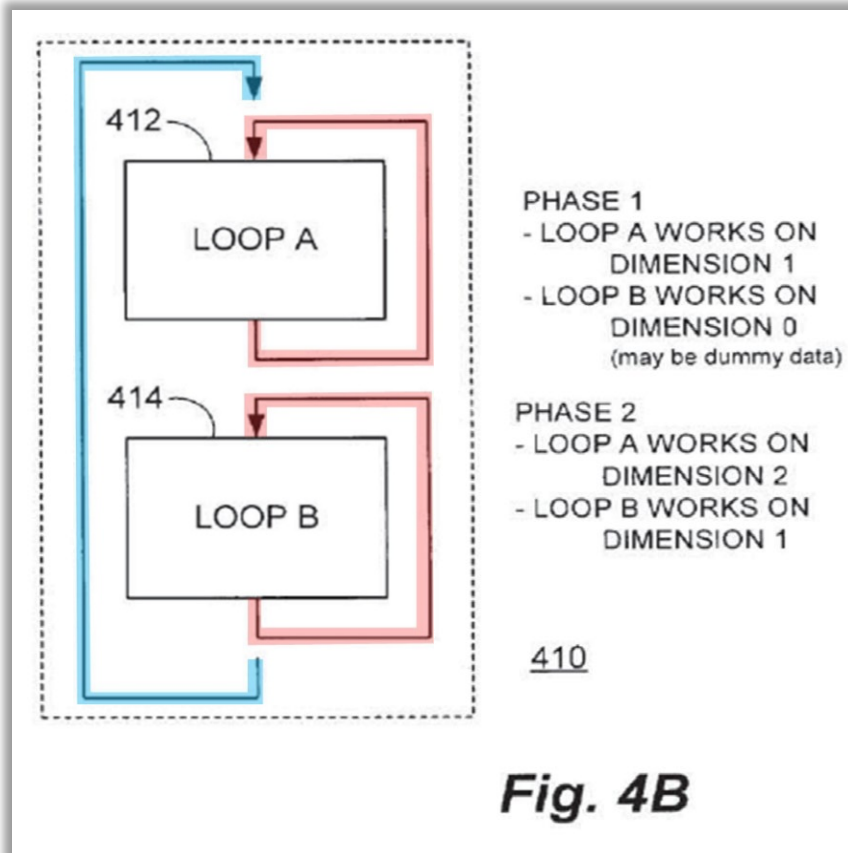


“computational loop”

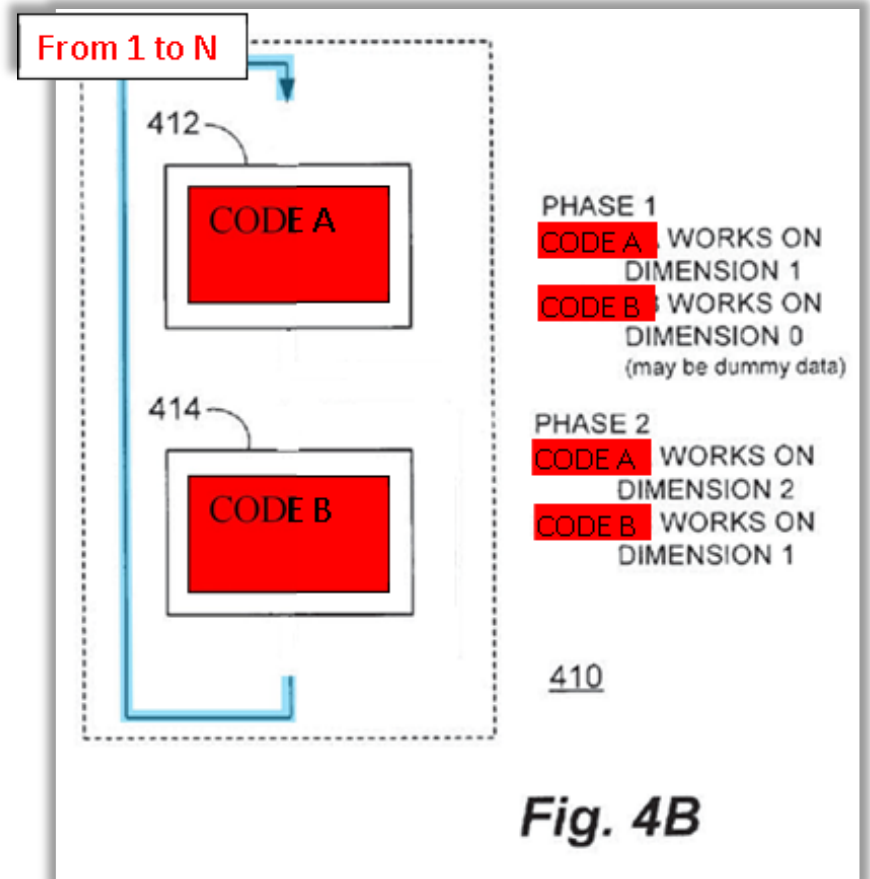
- Looping must be performed by the reconfigurable processor
 - As even Petitioner and its expert admit, instantiate means to “create, such as by configuring a particular structure.” Petition at 16-17; EX1003 ¶¶85-87.
 - Petitioner further referenced the file history that patentee stated “[a] reconfigurable processor is essentially a blank processor that must be configured (instantiated) to conduct a particular task... defining one particular variation of the processor’s structure.” 601 Petition at 17; EX1003 ¶¶85-87.
 - Thus, the parties agree that this instantiation requires configuring/instantiating the resources on the reconfigurable processor.

“computational loop”

- '324 Patent Teaching



- Splash2 Interpretation



“computational loop”

- Even Petitioner’s expert admitted that the pseudocode shown in Splash2 with the loop-endloop syntax would not be how the algorithm is built on an FPGA. Instead, the software would have to be converted using VHDL to instantiate distinct hardware, and “you don’t build them the way the software would read that.” EX2064 at 179:13-180:2.

17 THE WITNESS: I have already
18 testified that when you build these cells,
19 you don't build them the way the software
20 would read that. You would use VHDL and
21 convert this into parallel implementation.
22 So the cells that would be in the Splash 2
23 array would be the VHDL instantiation of
24 this loop, and each of those instantiations
25 would be the same loop, but they're
2 distinct, distinct hardware.

“computational loop”

- This is consistent with the looping to be handled by the Sun workstation rather than on the FPGAs of Splash2, which Petitioner’s expert admitted he has no personal knowledge about. EX2064 at 212:25-213:12.

Page 212	Page 213
1 STONE, Ph.D.	1 STONE, Ph.D.
2 Q. Okay. Are you familiar with what a	2 knowledge as to the design of the system? Were
3 hard macro is on an X -- or on a Xilinx FPGA?	3 you personally involved in the Splash 2 project
4 A. It's -- that's not exactly a term of	4 in any way?
5 art, but if you show me the document I would	5 A. No, I was not.
6 tell you if I recognize it or not. I mean, it	6 Q. How about any design of the
7 might be a --	7 place-and-route tools from Xilinx? Were you
8 Q. I don't know if I have one handy.	8 involved in the development of that?
9 A. It might be a Xilinx term. I can	9 A. Say again what the tools were?
10 conjecture what it means, but I -- I need the	10 Q. The place-and-route tools from
11 document to -- to be certain.	11 Xilinx. Did you have any personal involvement?
12 Q. But you don't recall reviewing any	12 A. No, I was not involved in that.
13 documentation or information about hard macros	13 Q. Now, on the Splash 2 system, would
14 for the Xilinx FPGA for this declaration?	14 you agree that the crossbar enables only a
15 A. Well, it comes to mind. I didn't	15 limited amount of general communication across
16 rely on it, and I think it might even be in the	16 the board?
17 Xilinx documents that I have cited, but I -- if	17 A. I have no way to say yes or no to
18 you show it to me, I will tell you whether I	18 that question. I have not studied it and not
19 recognize it or not.	19 prepared to answer it.
20 Q. I don't remember if I've asked you	20 Q. Okay. In terms of the -- I guess
21 this before.	21 the general routing network that's on the
22 Do you have any experience	22 Splash 2 board, does the -- does the Splash 2
23 programming on the Splash 2 system?	23 board -- actually, let me ask it this way:
24 A. I do not.	24 Does it provide some sort of a general routing
25 Q. Okay. And do you have any personal	25 network on the board itself?

“computational loop”

Dr. Stone also had no idea that Patent Owner did not rely upon or use certain “evidence” in its Response that was in fact first introduced by Petitioner in its Reply. *See* EX1074, and citations thereto in Reply. Instead, Petitioner launders this new exhibit and theories in its Reply via Dr. Stone by mischaracterizing Patent Owner’s evidence while falsely claiming Patent Owner relied on this exhibit to make opinions and arguments outside the scope of the Response. *See* EX2176 10:4-13:9 (discussing EX1074 and contents, first introduced by Petitioner in Reply, and outside the scope of the Patent Owner Response or evidence).

Finally, Dr. Stone did not attempt (nor was allowed to by Petitioner) to review the totality of Patent Owner’s evidence for his reply testimony, *see* EX2176 16:25-17:18, which is evidence contradicting his other opinions. *Compare* EX1076 ¶¶15-17, *with* EX1078 63:13-66:22 (Dr. El-Ghazawi explaining the irrelevance of the Gokhale article cited in the Halverson reference and how Halverson teaches how the Sun workstation in Splash2 shows no anticipation).

“computational loop”

[IPR01601: EX2064 at 176:13-177:25; IPR01605: EX2063 at 176:13-177:177:25].

192. A data register, unlike a FIFO, is an addressable storage location that would not allow data to “seamlessly” pass between PEs. Rather, a data register would require one PE to write data into it and the next PE to then read the data out. This is antithetical to the invention disclosed in the '324 and '800 patents.

193. A data register is not a “stream communication connection” and is not a “seamless” connection because one of the key innovations of the '324 and '800 patents is avoiding storing data to memory between computational loop (i.e., processing elements).

2. Splash2 does not teach computational loops.

194. The pseudo code shown in Figures 8.7 and 8.12 of Splash2 book, [EX1007 at 101 and 105], does not conform with a standard definition of a loop in computer programming.

195. The Board construed “computational loop” to mean “a set of computations that is executed repeatedly, either a fixed number of times or until some condition is true or false.”

196. The code shown in Figures 8.7 and 8.12 of Splash2 are not “computational loops.”

```
loop
  if (SCin ≠ 0) and (TCin ≠ 0) then
    PEDist ← min { PEDist+ψ(SCin,TCin),
                  TDin+ψ(SCin,0),
                  SDin+ψ(0,TCin) }
  else-if (SCin ≠ 0) then
    PEDist ← SDin
  else-if (TCin ≠ 0) then
    PEDist ← TDin
  endif
  SCout ← SCin
  TCout ← TCin
  SDout ← PEDist
  TDout ← PEDist
endloop
```

FIGURE 8.7 Code Executed by Each PE in the Bidirectional Array

```
loop
  if (TAGin = SR) then
    if (SRCch = 0) then
      SRCout ← CHRin
      CHROUT ← 0
      DSTout ← PDSTin
    else
      CHROUT ← CHRin
    endif
    PDSTout ← PDSTin
  else-if (TAGin = PR) then
    if (SRCch = 0) then
      DSTout ← PDSTin
    endif
    PDSTout ← DSTin
    CHROUT ← CHRin
  else-if (TAGin = TG) then
    if (SRCch ≠ 0) and (CHRin ≠ 0) then
      PDSTout+ψ(SRCch,CHRin),
      DSTout ← min { DSTin+ψ(SRCch,0),
                    DSTout+ψ(0,CHRin) }
    else-if (SRCch = 0) then
      DSTout ← DSTin
    endif
    PDSTout ← DSTin
    CHROUT ← CHRin
  endif
  TAGout ← TAGin
endloop
```

FIGURE 8.12 Code executed by each PE in the unidirectional array

197. Figures 8.7 and 8.12 each depict an “else if” conditional statement within the framework of a “loop-endloop.”

198. These conditional statements simply choose an execution path based on whether certain conditions are true or false. [EX2026 at 7 (Microsoft Dictionary definition of “conditional statement)].

199. For example, in Figure 8.7 the conditional statement first evaluates whether SCin and TCin are both not zero. Then based on whether that condition is true or false it selects an execution path.

“computational loop”

200. To be defined as a loop, a loop exit condition must be specified. This is generally specified either in the first line of the code (begin loop and then condition) or the last line of loop (loop end and then condition).

201. The pseudocode in Figures 8.7 and 8.12 does not specify an exit condition at the beginning or end.

202. So if these figures truly were loops, as Microsoft argues, then they would run forever since they lack an exit condition.

203. This means nothing would be calculated by the code because the first piece of data would enter each PE and then run forever. Nothing would be passed to the next PE. This is not how this code operates on Splash2.

204. The only reasonable interpretation of this pseudocode is to assume that something replaces the “loop-endloop” syntax so that the pseudocode executes once then passes the data to the next PE.

205. That is exactly what is depicted in Figure 8.6., which shows each PE executing the pseudocode once on the inputs and then sending the outputted data onto the next PE.

86

PATENT OWNER DIRECTSTREAM, LLC
EX. 2111, p. 89

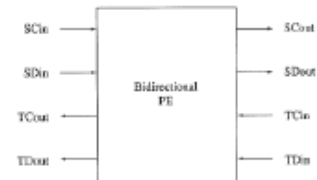


FIGURE 8.6 Processing Element for Bidirectional Array

206. None of the PEs in Splash2 evaluate any piece of data more than once

207. A computational loop evaluates each piece of data multiple times, “a fixed number of times or until some condition is true or false.”

208. Figures 8.7 and 8.12 would never evaluate the same data more than once so they are not computational loops.

209. Splash2 is not capable of handling loops independently. This is clearly discussed in Halverson: “When the actual program executes, looping is still handled in the Sun, which transmits SIMD instructions to the Splash 2 board(s).” [IPR01601: EX2167, page 14-15; IPR01605: EX2169, page 14-15]. Sun is the workstation that connects to Splash2 reconfigurable FPGAs and control them. Rather than Splash2 handles loops itself, the high performance workstation that controls the FPGAs compiles the loops, generates SIMD instructions and necessary signals (in forms of unrolled loops, for instance) for the FPGAs.

3. Splash2 requires memory between the PEs.

87

PATENT OWNER DIRECTSTREAM, LLC
EX. 2111, p. 90

“computational loop”

Page 10

1 counsel, I want to respond to also; is that correct?

2 A That's correct. I believe everything here
3 is relevant to computational loop.

4 Q Okay. Another little housekeeping matter.

5 In paragraph 2, I don't necessarily want to talk
6 about the sentence, but you see where it says

7 "EX2111"?

8 A I do.

9 Q It's exhibit number.

10 You have an understanding that's a piece of
11 evidence in this case and it's Exhibit 2111; do you
12 understand that?

13 A I do.

14 Q Okay. And then if we go to paragraph,
15 let's say, 4, where, again, you're talking about some
16 more verbiage and then you say "EX1001," that's,
17 again, some statement that you're supporting your
18 proposition by citing to Exhibit 1001; do you
19 understand that?

20 A I do.

21 Q At any time during these proceedings, it's
22 been going on for a while, did anyone ever explain to
23 you why some of them start with a 1 and some of the
24 exhibits start with a 2?

25 A Well, the 1s are exhibits that I produced

TSG Reporting - Worldwide - 877-702-9580

PATENT OWNER DIRECTSTREAM, LLC
EX. 2178, p. 10

Page 11

1 with my declaration or there may be additional ones
2 produced by Microsoft's counsel. The 2s are those
3 that were produced by the Patent Owners. That's my
4 understanding.

5 Q Okay. I just want to make sure. That's
6 correct. I want to make sure you understood that. A
7 lot of time it's not that I -- a lot of times people
8 are more interested in the citations and the EX
9 numbers usually the lawyers come back and write over
10 for clarity, okay. I just wanted to make sure you
11 understood that.

12 A Okay.

13 Q Can you turn to paragraph 15.

14 A I'm there.

15 Q Okay. This is starting section 2 of your
16 reply declaration called "Looping on Splash2". And
17 the paragraph reads "Patent Owner and its expert also
18 assert that the Splash2 system implemented loops on
19 an attached CPU rather than in the FPGAs of the
20 system relying solely on a paper by Gokhale" -- and I
21 think it's Minnich --

22 A It's --

23 Q -- entitled FPGA -- say again?

24 A I have the correct pronunciation. Gokhale
25 and Minnich.

TSG Reporting - Worldwide - 877-702-9580

PATENT OWNER DIRECTSTREAM, LLC
EX. 2178, p. 11

“computational loop”

1 Q -- "Gokhale and Minnich entitled FPGA
2 Computing In A Data Parallel C"; do you see that
3 sentence?

4 A I see that.

5 Q You use the word "solely" in there; do you
6 see that?

7 A I do.

8 Q Why?

9 A The support by Patent Owner and expert came
10 from Gokhale and Minnich in describing what happens
11 in that particular embodiment based on the data
12 parallel C. Other references have cited Gokhale and
13 Minnich, cited them for the support. So this is
14 where it came from.

15 Q And your support for that assertion is from
16 Patent Owner's response on page 88 and Patent Owner's
17 Exhibit 2111, paragraph 209; is that correct?

18 A That's correct.

19 Q Let's turn to paragraph 16. I'm not going
20 to read the whole paragraph, but I think you're
21 introducing the Gokhale reference at paragraph 16.
22 And when you flip the page from page 8 to page 9, you
23 introduce the Gokhale reference as Exhibit 1074; is
24 that correct?

25 A That's correct.

1 Q Okay. And we just mentioned a second ago
2 that if it has a 1, that's a Microsoft exhibit, not a
3 Patent Owner exhibit; is that correct?

4 A That's correct.

5 Q So I'm curious, how could Patent Owner have
6 solely relied on an exhibit that was actually
7 introduced by Microsoft in your reply?

8 A I can explain what happened from my end. I
9 don't have an answer to your question.

Disputed Claim Terms

- pass computed data “seamlessly” between said computational loops
- “systolic” and “data driven”
- “computational loop”
- **“stream communication”**

“stream communication” connection

B. “stream communication”

DirectStream’s Construction	Petitioner’s Construction
a data path that acts like a queue connecting via the reconfigurable routing resources a producer and a consumer of data that operate concurrently	Communication of a data sequence

“stream communication” connection

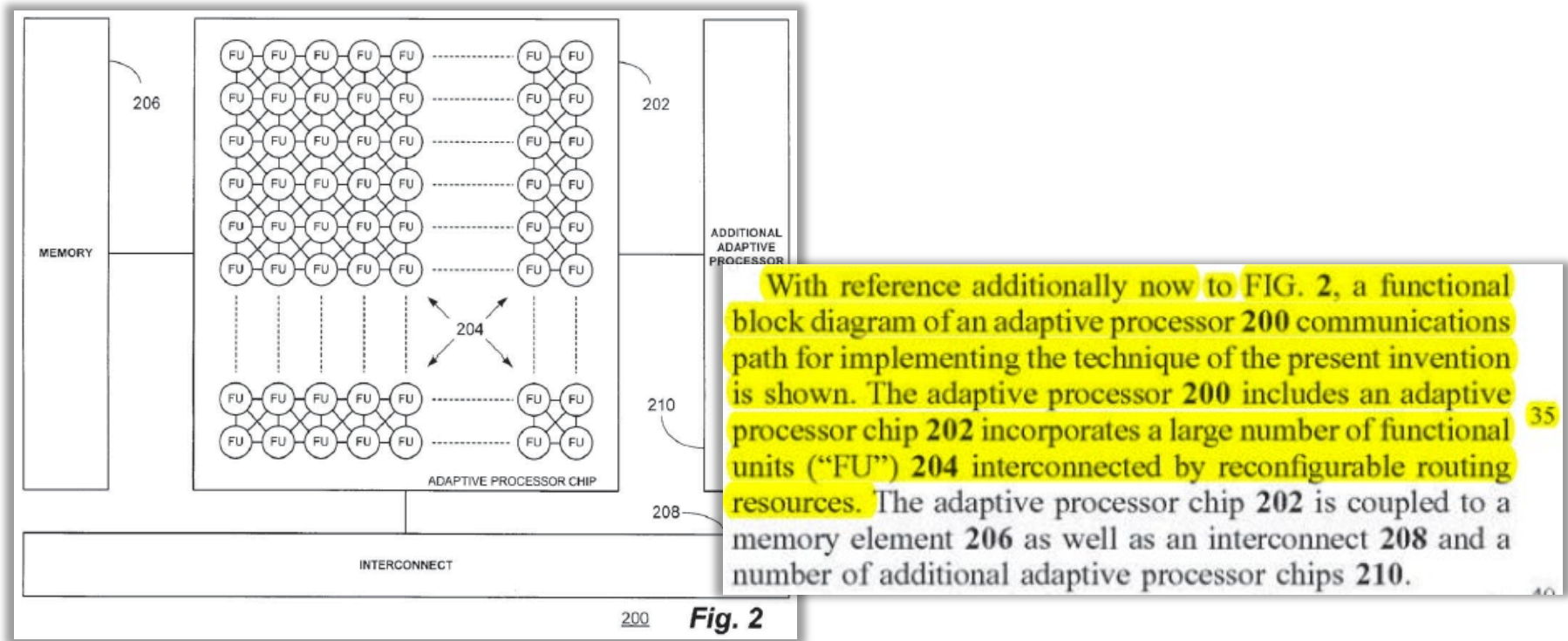
- The dependent claims for stream communication clearly state that the dependent claim narrows the “instantiation” limitation from the independent claims:
- 1. A method for data processing in a reconfigurable computing system, the reconfigurable computing system comprising at least one reconfigurable processor, the reconfigurable processor comprising a plurality of functional units, said method comprising:
 - ...
 - instantiating** at least two of said functional units at the at least one reconfigurable processor to perform said calculation wherein only functional units needed to solve the calculation are instantiated and **wherein each instantiated functional unit at the at least one reconfigurable processor interconnects with each other instantiated functional unit at the at least one reconfigurable processor based on reconfigurable routing resources within the at least one reconfigurable processor as established at instantiation, ...**
- 15. The method of claim 1 wherein **instantiating** includes establishing a stream communication connection **between functional units.**”

“stream communication” connection

- ...stream communication” defines what type of connection is being instantiated in dependent Claim 15
- The plain language of the claims requires
 - (1) instantiation of some structure to create a stream communication, and
 - (2) specific to the '324 Patent, this instantiation must utilize the reconfigurable resources within the reconfigurable processor. EX1001 at 12:63-13:8, 13:53-55.
- Stream communication must be instantiated to comply with a seamless, systolic implementation, per claim 1. EX1001 at 13:12-19.

“stream communication” connection

- Configured during instantiation to connect processing elements via the reconfigurable routing resources.



- EX1001 at 12:63-13:8, 5:31-53, Fig. 2; EX2111 ¶¶150-151.

“stream communication” connection

- As Petitioner and its expert admit, the specification does not provide a specialized definition of “stream communication” or otherwise offer any disclaimer of scope to alter the term from its plain and ordinary meaning. 601 Petition at 20; EX1003¶117.
- The intrinsic record does not unambiguously define “stream communication.” In fact, Petitioner’s own Petition conceded this: “The term ‘stream communication’ is not used in the ’324 Patent except in its claims, nor is it used in the incorporated references.” Petition, 20.
- The only allegedly supporting phrase referenced by Petitioner is different (stream of operands), which at best clarifies what is in the stream (operands), not the communication connection that must be instantiated (structure). It is also from a different patent referring to a different invention pertaining to internet communications.

“stream communication” connection


- The file history is equally unavailing.
 - The statement by the examiner regarding a “stream between processors” does not mention a queue. It also refers to the **contents** (“data is transferred systolically in at least one stream”), not to the **structure** of the stream communication connection that must be instantiated. Reply, 34.
 - Even Petitioner’s own brief concedes “the examiner noted...the claimed ‘stream communication connection’ [is] established as the interconnections are made,” but the examiner provides no statements on what specific structure is established. Reply, 34.
 - The examiner’s silence cannot “contradict” the extrinsic evidence; it is by definition silent one way or the other. If anything, there is consistency with the extrinsic evidence.

“stream communication” connection

- Because of the intrinsic record’s silence in defining a stream communication connection, Patent Owner provided numerous citations to references around the time of the invention that all consistently provide support for a POSITA’s understanding of the associated structure, consistent with Patent Owner’s proposed construction. Response, 53-62.
 - U.S. Patent No. 8,589,666 (assigned to Patent Owner and describing the prior art understanding of stream as a data path as shown in fig.1)
 - Patent Owner’s own product documentation³ and supporting declaration of the inventor of the ’324 Patent
 - Argonne National Laboratory article, defining stream as a data structure
 - U.S. Patent No. 5,748,613
 - European Patent No. 1820309
 - U.S. Patent No. 8,543,746
 - U.S. Patent No. 8,352,4564
 - U.S. Appl. No. 2010/0070730

“stream communication” connection

- EX2027 U.S. Patent No. 8,589,666
 - assigned to Patent Owner
 - describes the prior art understanding of stream as a data path as shown in fig.1


US008589666B2

(12) **United States Patent Hammes**

(10) **Patent No.:** US 8,589,666 B2
(45) **Date of Patent:** Nov. 19, 2013

(54) **ELIMINATION OF STREAM CONSUMER LOOP OVERSHOOT EFFECTS**

(75) **Inventor:** Jeffrey Hammes, Colorado Springs, CO (US)
(73) **Assignee:** SRC Computers, Inc., Colorado Springs, CO (US)

(*) **Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1985 days.

(21) **Appl. No.:** 11/456,466
(22) **Filed:** Jul. 10, 2006
(65) **Prior Publication Data**
US 2008/0010444 A1 Jan. 10, 2008

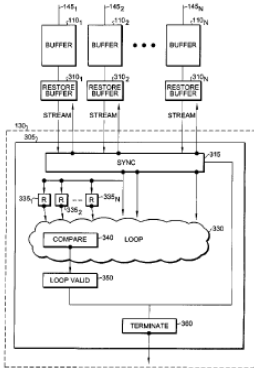
(51) **Int. Cl.**
G06F 15/00 (2006.01)
G06F 7/38 (2006.01)
G06F 9/00 (2006.01)
G06F 9/44 (2006.01)

(52) **U.S. Cl.** 712/241; 714/18
(58) **Field of Classification Search**
USPC 712/241, 201, 25
See application file for complete search history.

(56) **References Cited**
U.S. PATENT DOCUMENTS
7,085,955 B2 * 8/2006 Prabhu 714:6
2002/0124159 A1 * 9/2002 Bkooji et al. 712:226
* cited by examiner
Primary Examiner—Andrew Caldwell
Assistant Examiner—George Gironx
(74) *Attorney, Agent, or Firm*—William J. Kubisa; Hogan Lovells US LLP

(57) **ABSTRACT**
A reconfigurable processor invoking data stream pipelining is configured to associate a restore buffer with each incoming data stream. The buffer is configured to be of sufficient size to maintain data values dispatched to a loop so as to restore values fetched and lost due to loop overshoots. The restore buffer stores the values that were recently fetched from the buffer to the loop. To determine how many data values should be restored, the loop counts the number of the data values it takes from each data stream and the number of valid loop iterations that take place. Once a loop termination is detected, the loop halts the fetching of values from the restore buffer and compares, for each stream, the number of loop iterations with the number of values fetched. The difference is the number of extra values that were taken from the restore buffer and are restored.

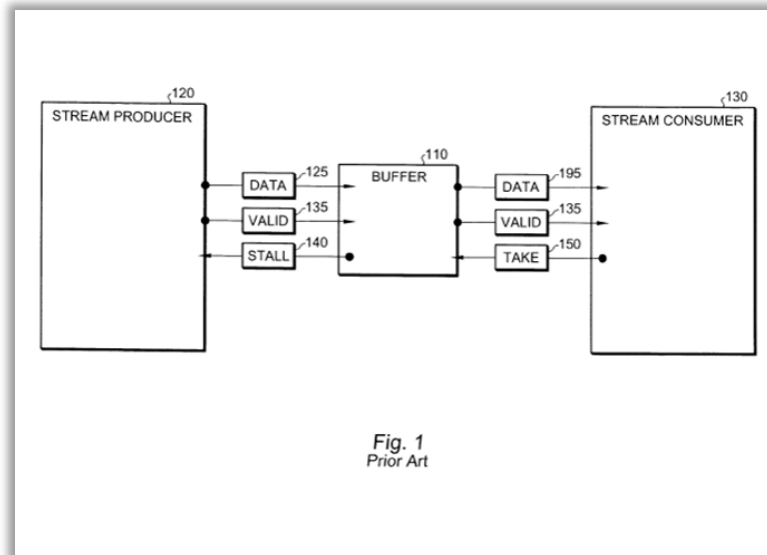
22 Claims, 6 Drawing Sheets



Patent Owner Saint Regis Mohawk Tribe
Ex. 2027 - p. 1

“stream communication” connection

- EX2027 at Fig. 1, 2:39-54:
 - “A stream is a data path between a producer and consumer of data, where the producer and consumer run concurrently. The path between the producer and consumer is made up of a data connection, a “valid” signal, and a reverse direction “stall” signal. FIG. 1 shows typical signals used in a stream connection as is well known and will be recognized by one skilled in the relevant art.”



“stream communication” connection

- EX2027 at 2:45-63.

“The use of a First-In-First-Out buffer 110, or “FIFO” buffer, removes the need for tight synchronization between the producer 120 and consumer 130. The producer 120 will generate data values 125 at its own rate, allowing them to accumulate in the FIFO buffer 110. As the FIFO buffer 110 approaches becoming full, it will issue a stall signal 140 to the producer 120 so that it will suspend the generation of data values 125 until the stall signal is released. The consumer 130 will take 150 values 145 from the FIFO buffer at its own rate and as the values 145 are available.

The use of the FIFO buffer, with the valid 135, stall 140 and take 150 signals, allows flexible coupling of stream producers and consumers. A stream's producer 120 and its consumers 130 may run at different speeds. For example, when the producer 120 runs faster than the consumer 130, then it will stall 140 from time to time as values fill the FIFO buffer. When the producer runs slower than the consumer, the FIFO will sometimes be empty and the consumer will wait for new values to be available.”

- EX2027 at 6:6-13:

“As previously described, a typical configuration of a pipelined data stream structure places a single buffer between the producer and the consumer. Such a single buffer is typically configured to absorb differences in the production and consumption rate of the data streams, but does little to prevent the lost of data due to consumer loop overshoots.”

“stream communication” connection

- DirectStream’s ’666 Patent describes as part of the technical background the use of a FIFO buffer to communicate between two processing elements, in particular where the producer processor can add data to the FIFO buffer at its rate and the consumer processor can remove data from the FIFO buffer at its separate rate. EX2111 ¶¶152-154, 179-187. The disclosed use of a FIFO buffer structure for stream communication allows the producer and consumer to run concurrently, while the buffer absorbs differences in the production and consumption rate of the data streams. EX2111 ¶¶152-157, 179-187.
- This disclosure is entirely consistent with the ’324 Patent’s teachings for instantiating reconfigurable resources to seamlessly communicate computed data between processing elements in independent claim 1, and more particularly to provide stream communication between those processing elements in the dependent claim 15. EX2111 ¶¶152-157, 179-187; EX1001 at 12:63- 13:8, 13:53-55.

“stream communication” connection

SRC Carte™ C Programming Environment v3.0 Guide (Pre-Release)



Copyright © 2002 - 2007 SRC Computers, Inc.
ALL RIGHTS RESERVED.

November 20, 2007
SRC-007-19

PATENT OWNER DIRECTSTREAM, LLC
EX. 2107, p. 2

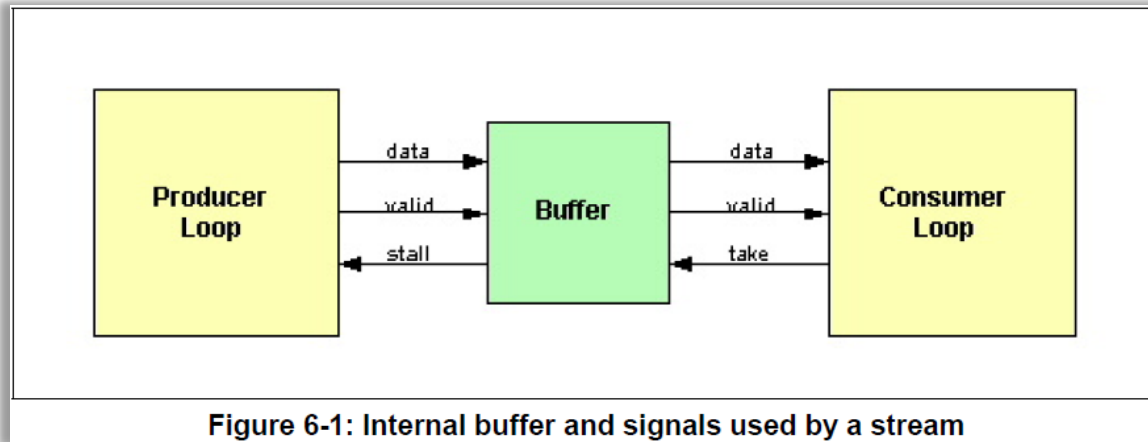
- EX2107 Patent Owner's product documentation

“stream communication” connection

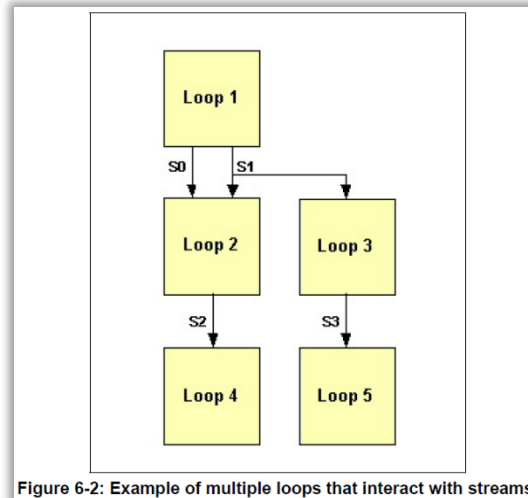
- DirectStream’s own product documentation describes a stream as a data structure that allows flexible communication between concurrent producer and consumer loops, which is consistent with how a POSITA would understand this term in the context of the claims, particularly as part of instantiating structure on a reconfigurable processor. EX2100 ¶¶79; EX2107 at 94-98; EX2111 ¶¶150-154, 182- 187.

“stream communication” connection

- EX2107, Fig. 6-1 at p. 94,



- EX2107, Fig. 6-2 at 96



“stream communication” connection

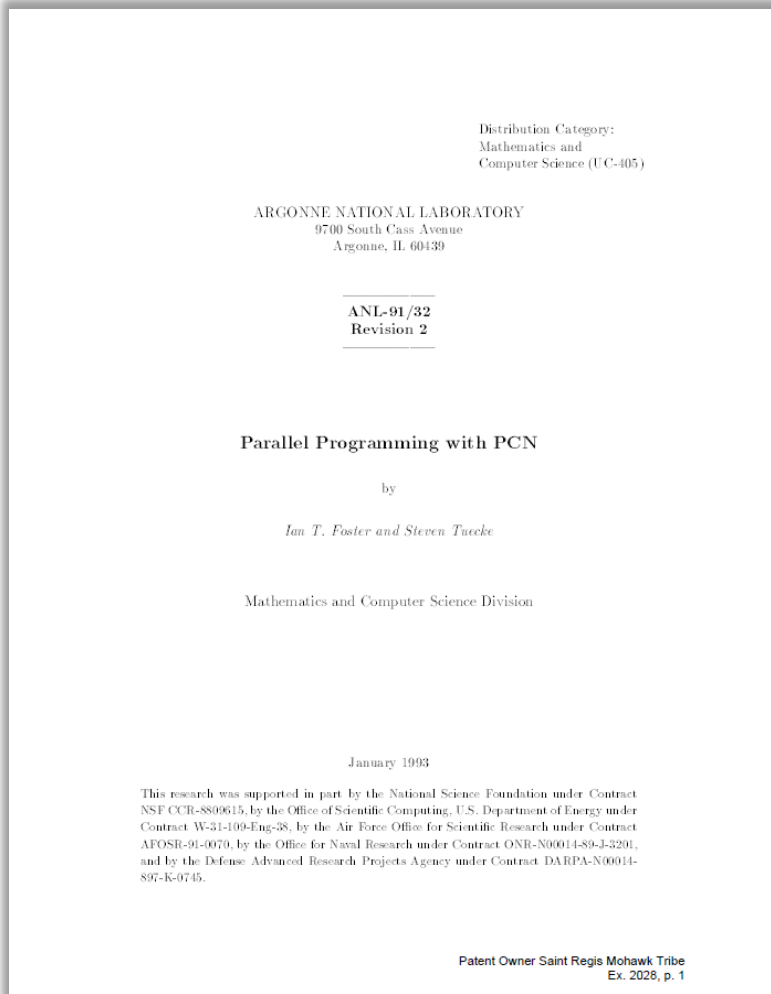
- POSITA would recognize that a queue is a well-known data structure with first-in-first-out properties in which only two actions are permitted, (i) the addition of entities to the rear position, known as enqueueing, and (ii) the removal of entities from the front position, known as dequeueing. This ensures that data entering a queue remains in the same order that it arrived.

queue¹ *n.* A multi-element data structure from which (by strict definition) elements can be removed only in the same order in which they were inserted; that is, it follows a first in, first out (FIFO) constraint. There are also several types of queues in which removal is based on factors other than order of insertion—for example, some priority value assigned to each element. *See also* deque, element (definition 1). *Compare* stack.

- EX2065 at 433; EX2111 ¶¶152-154, 182-185.

“stream communication” connection

- EX2028 Argonne National Laboratory article
 - defines stream as a data structure



“stream communication” connection

- A paper from Argonne National Laboratory from 1993 defines “stream” as follows:

4.9 Stream Communication

We have seen how two or more concurrent computations that share a definitional variable can use that variable to exchange data. The *producer* of the data simply defines the shared variable to be the data to be communicated (e.g., $x = \text{"hello"}$). The *consumer(s)* of the data can then use the data in computation.

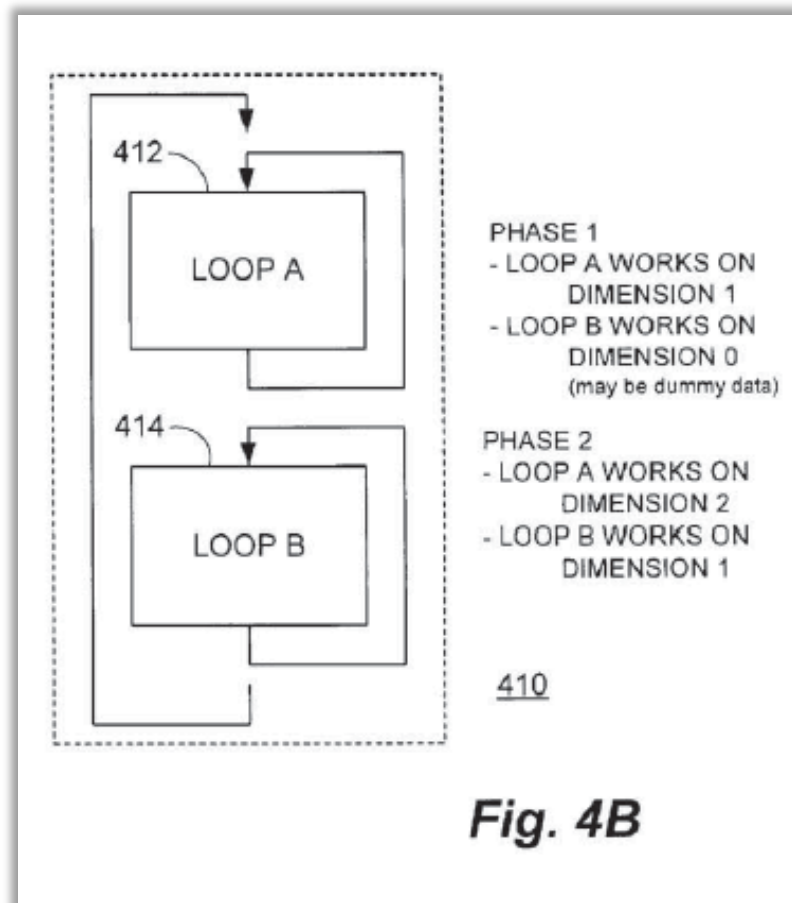
A shared definitional variable would not be very useful if it could be used only to exchange a single value. Fortunately, there are simple techniques that allow a single definitional variable to be used to communicate many values. The most important of these is the *stream*. A stream is a data structure that permits communication of a sequence of messages from a producer to one or more consumers. A stream acts like a queue: the producer places elements on one end, and the consumer(s) take them off the other.

By convention, stream communication is implemented in PCN in terms of list structures. Imagine a producer and a consumer sharing a variable x . The producer defines $x = [\text{msg}|x\tau]$ and the consumer matches $x \text{ ?} = [\text{msg}|x\tau]$. The effect of these operations is to both communicate msg to the consumer and create a new shared variable $x\tau$ that can be used for further communication. This process can be repeated arbitrarily often to communicate a stream of messages from the producer to the consumer. Hence, a stream is a list structure, incrementally constructed by a producer and deconstructed by a consumer. The empty list ($[\]$) is used to represent the end of a stream.

- EX2028 at 31.

“stream communication” connection

- This is consistent with the '324 patent's use of a “stream communication” to connect the functional units that form the two computational loops.
EX1001 at 13:53-55;
EX2111 ¶¶ 150-154, 179-187.
- Those two loops act as producers and consumers with the stream communication enabling the data to pass from the producer (loop 1) to consumer (loop 2).



“stream communication” connection

- This same concept is also consistently demonstrated by other third party patents that both pre- and post-date the '324 Patent
 - EX2169 - U.S. Patent No. 5,748,613
 - EX2170 - European Patent No. 1820309
 - EX2171 - U.S. Patent No. 8,543,746
 - EX2172 - U.S. Patent No. 8,352,456
 - EX2173 - U.S. Appl. No. 2010/0070730

"stream communication" connection

US005748613A

United States Patent [19]
Kilk et al.

[11] **Patent Number:** 5,748,613
 [45] **Date of Patent:** May 5, 1998

[54] **COMMUNICATION PACING METHOD**

[75] **Inventors:** Erik Kilk, Battleground; Karen Vander Veer, Vancouver, both of Wash.; Leann M. MacMillan, West Linn, Oreg.

[73] **Assignee:** Hewlett-Packard Company, Palo Alto, Calif.

[21] **App. No.:** 626,224
 [22] **Filed:** Mar. 29, 1996

[51] **Int. Cl.⁶** G06F 11/00; H04L 12/56
 [52] **U.S. Cl.** 370/231; 370/236; 370/473; 395/200.62

[58] **Field of Search** 395/200.62, 200.63; 370/231, 235, 236, 465, 473

[56] **References Cited**

U.S. PATENT DOCUMENTS



4,543,644	9/1985	Kozima et al.	364/900
4,830,851	6/1989	Kobayashi et al.	370/231
5,123,061	6/1992	Patchard	382/56
5,432,784	7/1995	Oveeren	370/235
5,482,824	7/1995	Zheng et al.	370/356
5,453,982	9/1995	Pennington et al.	370/85.1
5,515,359	5/1996	Zheng	370/231
5,528,591	6/1996	Lauer	370/231
5,633,867	5/1997	Ben-Nun et al.	370/399

18 Claims, 4 Drawing Sheets

PATENT OWNER DIRECTSTREAM, LLC
 EX. 2169, p. 1

- EX2169 - U.S. Patent No. 5,748,613
- EX2169 at Abstract: The present invention provides a method of pacing a stream of data transmitted from a data source to a buffered data destination with a determined number of available storage units, the data destinations being configured to consume data and thereby to free storage units for receipt of additional data. The pacing of data communication includes: (1) identifying a beginning credit value; (2) incrementing the beginning credit value with each storage unit freed to identify an present credit value; (3) transmitting units of data in accordance with determined limits, the number of data units sent providing a transmission count; (4) selectively updating the determined number of available storage units by determining the difference between the beginning credit value and the present credit value, and determining the sum of the result and the previously determined number of available storage units to provide an updated determined number of available storage units; and (5) selectively updating the determined number of available storage units by determining the difference between the transmission count and the previously determined number of available storage units to provide an updated determined number of available storage units.
- EX2169 at 3:31-35: Referring still to FIG. 1, it will be understood that data producer 12 typically includes a memory and a processor capable of providing an image buffer 16, a command protocol buffer 18, an auto-status buffer 20, a device ID buffer 22, and a pacing buffer 24.

“stream communication” connection

(19)  (11)  EP 1 820 309 B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent: 27.08.2008 Bulletin 2008/35 (51) Int. Cl.: H04L 12/56 (2006.01)

(21) Application number: 05850071.1 (86) International application number: PCT/IB2005/053970

(22) Date of filing: 30.11.2005 (87) International publication number: WO 2006/059283 (08.06.2006 Gazette 2006/23)

(54) **STREAMING MEMORY CONTROLLER**
STREAMING-SPEICHERSTEUERUNG
CONTROLEUR DE MEMOIRE EN CONTINU

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR
HU IE IS IT LI LT LU LV MC NL PL PT RO SE SI
SK TR
Designated Extension States:
AL BA HR MK YU

(86) HEKSTRA-NOWACKA, Ewa
NL-5656 AA Eindhoven (NL)
• HARMSZÉ, Françoise, J.
NL-5656 AA Eindhoven (NL)
• VAN DEN HAMER, Peter
NL-5656 AA Eindhoven (NL)

(30) Priority: 03.12.2004 EP 04106274 (74) Representative: van der Veer, Johannes Leendert
et al
NXP Semiconductors B.V.
IP&L Department
High Tech Campus 32
5656 AE Eindhoven (NL)

(43) Date of publication of application: 22.08.2007 Bulletin 2007/34

(73) Proprietor: Koninklijke Philips Electronics N.V.
5621 BA Eindhoven (NL)

(56) References cited:
US-A- 5 751 951 US-A1- 2002 034 162
US-B1- 6 405 256

(72) Inventors:
• BURCHARD, Artur
NL-5656 AA Eindhoven (NL)

EP 1 820 309 B1

Note: Within nine months of the publication of the mention of the grant of the European patent in the European Patent Bulletin, any person may give notice to the European Patent Office of opposition to that patent, in accordance with the Implementing Regulations. Notice of opposition shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

Printed by Jouve, 75001 PARIS (FR)

PATENT OWNER DIRECTSTREAM, LLC
EX. 2170, p. 1

- EX2170 – EP No. 1820309
- EX2170 at 0003-0004.
- [0003] Buffering is essential in a proper support of data streaming between the involved processes. Typically, FIFO buffers are used for streaming, which is in accordance to (bounded) Kahn process network models of streaming application. With increased number of multimedia applications that can run simultaneously the number of processes, realtime streams, as well as the number of associated FIFOs, substantially increases.
- [0004] There exist two extreme implementations of streaming with respect to memory usage and FIFOs allocation. The first uses physically distributed memory, where FIFO buffers are allocated in a local memory of a subsystem. The second uses physically and logically unified memory where all FIFO buffers are allocated in a shared, often off-chip, memory. A combination thereof is also possible.

“stream communication” connection

- EX2171 - U.S. Patent No. 8,543,746
- EX2171 at Abstract.
- A circuit arrangement and method facilitate the direct streaming of data between producer and consumer circuits (12P, 12C) that are otherwise configured to communicate over an address-based network (18). Sync signals (46,56) are generated for each of producer and consumer circuits (12P, 12C) from the address information encoded into requests that communicate the data streams output by the producer circuit (12P) and expected by the consumer circuit (12C). The sync signals (46,56) for the producer and consumer circuits (12C) are then used to selectively modify the data stream output by the producer circuit (12P) to a format expected by the consumer circuit (12C). Typically, such modification takes the form of inserting data into the data stream when the consumer circuit (12C) expects more data than output by the producer circuit (12P), and discarding data communicated by the producer circuit (12P) when the consumer expects less data than that output by the producer circuit (12P).

US08543746B2

(12) **United States Patent** (10) **Patent No.:** **US 8,543,746 B2**
Roever (45) **Date of Patent:** **Sep. 24, 2013**

(54) **SEL-SYNCHRONIZING DATA STREAMING BETWEEN ADDRESS-BASED PRODUCER AND CONSUMER CIRCUITS** (58) **Field of Classification Search**
USPC 710/29; 58; 60; 61; 100; 306; 308; 710/310; 713/373; 400; 401
See application file for complete search history.

(75) **Inventor:** **Jens Roever**, Los Gatos, CA (US)
(73) **Assignee:** **NXP B.V.**, Eindhoven (NL)

(*) **Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 748 days.

(21) **App. No.:** **11/917,624**
(22) **PCT Filed:** **Jun. 25, 2006**
(86) **PCT No.:** **PCT/IB2006/052068**
§ 371(a)(1),
(2), (4) **Date:** **Jul. 7, 2009**
(87) **PCT Pub. No.:** **WO2006/137044**
PCT Pub. Date: **Dec. 28, 2006**

(65) **Prior Publication Data**
US 2009/0300256 A1 Dec. 3, 2009

Related U.S. Application Data
(60) **Provisional application No. 60/594,113**, filed on Jun. 24, 2005.

(51) **Int. Cl.**
G06F 1/00 (2006.01)
G06F 3/00 (2006.01)
G06F 5/00 (2006.01)
G06F 13/06 (2006.01)
G06F 1/04 (2006.01)
G06F 1/12 (2006.01)
G06F 15/16 (2006.01)
G06F 13/12 (2006.01)
H04L 5/00 (2006.01)
H04L 7/00 (2006.01)

(52) **U.S. Cl.**
USPC 710/100; 710/29; 710/58; 710/60;
710/61; 710/306; 710/308; 710/310; 713/373;
713/400; 713/401


33 Claims, 7 Drawing Sheets

The diagram shows a data stream path from a producer circuit to a consumer circuit. On the left, a 'PRODUCER CIRCUIT' (12P) includes an 'IP BLOCK PRODUCER' (14) and a 'REQUEST' (16) block. The data path goes through a 'PRODUCER SYNC' block (30) and a 'STREAM CTRL' block (32). On the right, a 'CONSUMER CIRCUIT' (12C) includes a 'CONSUMER SYNC' block (40) and an 'IP BLOCK CONSUMER' (14). The data path goes through a 'REQUEST' (16) block. A 'SYNC SIGNAL' (46, 56) is shown as a control signal between the producer and consumer sync blocks. The diagram is labeled with various reference numerals (14, 16, 24, 26, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56).

PATENT OWNER DIRECTSTREAM, LLC
EX. 2171, p. 1

“stream communication” connection

- EX2172 - U.S. Patent No. 8,352,456
- EX2172 at 1:29-50.
- One basic design pattern is producer/consumer. A producer/consumer relationship is one in which a producer generates data and the consumer uses the data. This pattern is utilized in a myriad of different environments for a number of processes including, at a higher level, data warehousing for cleansing and transforming data and image processing for iterative refinement. In fact, the pattern can apply to any situation in which data is produced and consumed. One particularly prevalent use case pertains to queries. Query execution can be seen as a traditional client/server or consumer/producer model where an entity A requests a service from another entity B, in this case the retrieval of some data that satisfies criteria and is in the shape requested. Some bi-directional communication mechanism is required such that A can instruct B about its desire and so that B may respond to A with the results. The entire result set is returned in some form and thereafter consumed for some purpose. ...



US008352456B2

(12) **United States Patent**
Duffy et al.

(10) **Patent No.:** US 8,352,456 B2
(45) **Date of Patent:** Jan. 8, 2013

(54) **PRODUCER/CONSUMER OPTIMIZATION**
(75) **Inventors:** John J. Duffy, Renton, WA (US); Henrikus Johannes Maria Meijer, Mercer Island, WA (US)

(73) **Assignee:** Microsoft Corporation, Redmond, WA (US)

(*) **Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 590 days.

(21) **Appl. No.:** 11/747,772
(22) **Filed:** May 11, 2007

(65) **Prior Publication Data**
US 2008/0281786 A1 Nov. 13, 2008

(51) **Int. Cl.**
G06F 7/00 (2006.01)
G06F 17/30 (2006.01)

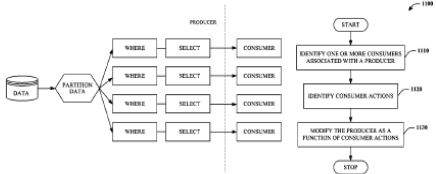
(52) **U.S. Cl.**
707/713; 707/719

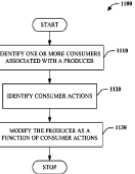
(58) **Field of Classification Search**
707/713; 707/719; 999.002
See application file for complete search history.

(56) **References Cited**
U.S. PATENT DOCUMENTS
5,590,319 A 12/1996 Cohen et al.
5,577,110 A 1/1999 Hallmark et al.
6,099,265 A 12/1999 Huang et al.
6,825,593 B1 8/2003 Long et al.
6,968,335 B2 11/2005 Bayliss et al.
7,051,024 B1 5/2006 Oishi et al.
7,146,365 B2 12/2006 Allen et al.
2005/0165798 A1 7/2005 Cherkauer et al.

OTHER PUBLICATIONS
Ganguly et al., "Query Optimization for Parallel Execution", Proceedings of the 1992 ACM SIGMOD international conference on Management of data, p. 9-18, Jun. 2-5, 1992, San Diego, California, United States.*

12 Claims, 13 Drawing Sheets





PATENT OWNER DIRECTSTREAM, LLC
EX. 2172, p. 1

“stream communication” connection

US 20100070730A1

(19) **United States**
(12) **Patent Application Publication** (10) **Pub. No.: US 2010/0070730 A1**
Pop et al. (43) **Pub. Date: Mar. 18, 2010**

(54) **MINIMIZING MEMORY ACCESS CONFLICTS OF PROCESS COMMUNICATION CHANNELS**
Publication Classification
(51) **Int. Cl.** G06F 12/00 (2006.01)
(52) **U.S. Cl.** 711/167; 711/E12.001
(57) **ABSTRACT**
A system and method for minimizing cache conflicts and synchronization support for generated parallel tasks within a compiler framework. A compiler comprises library functions to generate a queue for parallel applications and divides it into windows. A window may be sized to fit within a first-level cache of a processor. Application code with producer and consumer patterns within a loop construct has these patterns split into producer and consumer tasks. Within a producer task loop, a function call is placed for a push operation that modifies a memory location within a producer sliding window without a check for concurrent accesses. A consumer task loop has a similar function call. At the time a producer or consumer task is ready to move, or slide, to an adjacent window, its corresponding function call determines if the adjacent window is available.

Correspondence Address:
MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL (AMD)
P.O. BOX 598
AUSTIN, TX 78767-0598 (US)

(21) Appl. No.: 12/212,370
(22) Filed: Sep. 17, 2008

PATENT OWNER DIRECTSTREAM, LLC
EX. 2173, p. 1

- EX2173 - U.S. Appl. No. 2010/0070730
- EX2173 at 0012. In one embodiment, a method comprises dividing a stream into windows, wherein a stream is a circular first-in, first-out (FIFO) shared storage queue. In one window, a producer task is able to modify memory locations within a producer sliding window without checking for concurrent accesses to the corresponding elements.
- EX2173 at 0044. The transfer of data between tasks as shown in FIG. 3A-3C may occur via a communication channel called a stream. A stream may be a circular buffer managed as a FIFO concurrent lock free queue. Concurrent FIFO queues are widely used in parallel applications and operating systems. A stream may be implemented in the memory hierarchy 400 such as in stream copies 440 and 460. The most updated contents of a stream may be in Stream copies located closest to a processor core, such as stream copy 440.

“stream communication” connection

- The foregoing extrinsic references are still consistent with the intrinsic record.
- Within the context of the '324 Patent, a POSITA would recognize that this inter-chip communication concept can be adapted to intra-chip communications between functional units on the same chip—e.g., a FIFO within the chip:

As shown, each adaptive processor chip 202 can contain thousands of functional units 204 dedicated to the particular problem at hand. Interconnect between these functional units is created by reconfigurable routing resources inside each chip 202. As a result, the functional units 204 can share or exchange data at much higher data rates and lower latencies than a standard microprocessor 104 (FIG. 1). In addition, the adaptive processor chips 202 can connect directly to the inter-processor interconnect 208 and do not require the data to be passed through multiple chips in a chipset in order to communicate. This is because the adaptive processor can implement whatever kind of interface is needed to accomplish this connection.

- EX1001 at 5:41-53.

“stream communication” connection

- The specification discloses neighboring cell communications and the use of scheduling to eliminate the need for data storage, as well as the concept of using chain ports and a FIFO buffer for chip to chip communications, borrowing concepts from the '819 Patent owned by the same applicant.

As shown, the computation of fluid flow properties are communicated to neighboring cells 710 and, importantly, this computation can be scheduled to eliminate the need for data storage. In accordance with the technique of the present invention, a set of cells can reside in an adaptive processor and the pipeline of computation can extend across multiple adaptive processors. Communication overhead between multiple adaptive processors may be advantageously minimized through the use of MAP™ adaptive processor chain ports as disclosed in U.S. Pat. No. 6,339,819 issued on Jan. 15, 2002 for: “Multiprocessor With Each Processor Element Accessing Operands in Loaded Input Buffer and Forwarding Results to FIFO Output Buffer”, assigned to SRC Computers, Inc., assignee of the present invention, the disclosure of which is herein specifically incorporated by this reference.

- EX1001 at 7:59-8:6.

“stream communication” connection

- The '687 Patent is related to the '819 Patent, which the '324 Patent incorporated by reference, and it further describes the chain port:

10 With reference additionally now to FIG. 7, an additional functional block diagram of an individual MAP element 112 is shown particularly illustrating the memory 40 of the preceding figure functioning as an input buffer 40 and output FIFO 74 portions thereof. In this figure, an alternative view of the MAP element 112 of FIG. 6 is shown in which memory input data on line 50 (or the write trunk 26) is supplied to an input buffer (memory 40) as well as to a reconfigurable user array 42 coupled to the chain port 24.

15 The output of the reconfigurable array 42 is supplied to an output FIFO 74 to provide memory output data on line 94 (or the read trunk 28) as well as to the chain port 24. The input buffer 40, reconfigurable array 42 and output FIFO 74 operate under the control of the control block 46.

20 With respect to the foregoing figures, each MAP element 112 may consist of a printed circuit board containing input operand storage (i.e. the memory/input buffer 40), user array 42, intelligent address generator control block 46, output result storage FIFO 74 and I/O ports to allow connections to

25 other MAP elements 112 through the chain port 24 as well as the host system memory array.

- EX1014 at 8:7-26.

“stream communication” connection

- Additionally, alongside the chain port, the '687 Patent discloses input and output FIFO buffers as part of chip to chip communication, which relates to the '324 Patent claim for stream communication between functional units on the same chip to solve the technical problem. See EX2111 ¶¶125-131, 150-152.

10 With reference additionally now to FIG. 7, an additional functional block diagram of an individual MAP element 112 is shown particularly illustrating the memory 40 of the preceding figure functioning as an input buffer 40 and output FIFO 74 portions thereof. In this figure, an alternative view of the MAP element 112 of FIG. 6 is shown in which memory input data on line 50 (or the write trunk 26) is supplied to an input buffer (memory 40) as well as to a reconfigurable user array 42 coupled to the chain port 24. 15 The output of the reconfigurable array 42 is supplied to an output FIFO 74 to provide memory output data on line 94 (or the read trunk 28) as well as to the chain port 24. The input buffer 40, reconfigurable array 42 and output FIFO 74 operate under the control of the control block 46.

- EX1014 at 8:7-26.

“stream communication” connection

- And continuing:

The array 42 performs the actual computational functions of the MAP element 112. It may comprise one or more high performance field programmable gate arrays (“FPGAs”) interconnected to the other elements of the MAP element 112. A particular implementation of the present invention disclosed in more detail hereinafter, may use four such devices yielding in excess of 500,000 usable gates. These components are configured by user commands that load the contents of selected configuration ROMs into the FPGAs. After configuration, the user array 42 can perform whatever function it was programmed to do. In order to maximize its performance for vector processing, the array 42 should be able to access two streams of operands simultaneously. This is accomplished by connecting one 72 bit wide input port to the input operand storage and a second 72 bit wide port to the chain input connector port 24. This connector allows the MAP element 112 to use data provided to it by a previous MAP element 112. The chain port 24 allows functions to be implemented that would far exceed the capability of a single MAP element 112 assembly. In addition, since in the particular implementation shown, only operands are transferred over the chain port 24, the bandwidth may exceed the main memory bandwidth resulting in superior performance to that of the fixed instruction microprocessor-based processors 12.

When the user array 42 produces a result, it may be sent over a 72 bit wide path to an output result storage element (for example, output FIFO 74) which can then pass the data to either a 72 bit wide read port or a 72 bit wide chain port 24 to the next MAP element 112. This storage device can be made from a number of different memory types. The use of a FIFO 74 storage device will temporarily hold results that cannot be immediately read by a host microprocessor or passed over the output chain port 24 to the next stage. This feature allows for MAP elements 112 in a chain to run at different frequencies. In this case the output FIFO 74 functions like a speed matching buffer. In non-chained operation, the microprocessor that is reading the results may be delayed. In this case the FIFO 74 prevents the MAP element 112 from “stalling” while waiting for results to be read. In a particular embodiment of the present invention, a FIFO 74 that is 72 bits wide and 512K entries deep may be utilized. As disclosed in the aforementioned patent applications, the output storage may also be a true memory device such as those found in common memory. In this case, write addresses must be provided by the user array 42 or address generator and read addresses provided by the entity reading the results from the memory. While this may be somewhat more electrically complicated, it has the advantage that results may be accessed in any order.

- EX1014 at 9:2-67.

“stream communication” connection

- Petitioner’s own expert concurs the patent excludes storing values in memory—e.g., sending data off the chip.

Page 85

1 STONE, Ph.D.
2 Oh, there it is. Okay.
3 So this concept of systolic, what's
4 your opinion of what that means?
5 A. My opinion of what it means is in
6 the report. It means it's the characteristic
7 of rhythmically computing and passing data
8 directly between processing elements. And then
9 I quote: "Without a program counter or clock
10 that drives a movement of data," and also
11 operating in a manner that is, "transport
12 triggered, i.e., by the arrival of a data
13 object."
14 Q. Okay. And you mention the word
15 "directly," it was passing data directly
16 between processing elements. What does that
17 phrase mean to you or what's the context?
18 A. What are you trying to describe
19 there?
20 A. That the data goes from first to the
21 second without going to something intervening.
22 It directly go -- is connected immediately.
23 Indirectly we -- you go through one or more
24 intervening places to get there.
25 Q. Okay. So would memory, if the data

Page 86

1 STONE, Ph.D.
2 was going from one processing element to memory
3 and then back to a processing element, is that
4 something you would consider as an intervening
5 thing?
6 A. Well, that would not be a direct
7 connection of the output of the cell to the
8 next cell. It says, "Between processing
9 elements you're directly connected." If you're
10 saying you have a processing element outputting
11 to memory and then coming back to another
12 processing element, that would not be direct.
13 Q. Okay. Are there any other examples
14 of intervening structures or circuits that
15 would violate this direct connection?
16 A. I -- I think you're opening a
17 universe. I'm not going to answer that because
18 I'd like to -- let's get specific things.
19 Q. Well, how about a -- a register?
20 Would that be an intervening structure?
21 A. I -- I'm puzzled because that --
22 that register would be within -- within the
23 processing element in my mind.
24 Q. Okay.
25 A. If it's within the processing

- EX2064 at 85:14-86:12. See also EX1003¶117

“stream communication” connection

- Petitioner’s construction is flawed under any claim construction standards because it
 - 1) results in an illogical definition that destroys the independent-dependent relationship of the claims,
 - 2) improperly broadens the term so as to strip it of all meaning relative to the rest of the claim language, and
 - 3) is inconsistent with POSITA’s understanding of the plain and ordinary meaning.

“stream communication” connection

- Petitioner ignores the canon of claim construction that a dependent claim scope should be differentiated from its independent claim.
- Petitioner’s proposed definition of a “communication of data sequence” is already present in any data processing systems to begin with—including in a systolic data processing system—adding no limitations that are not already in the independent claims.
- As such, Petitioner’s proposed definition would violate claim differentiation by rendering the dependent claims meaningless and of the same exact scope as the independent claim. *Am. Piledriving Equip., Inc. v. Geoquip, Inc.*, 637 F.3d 1324, 1335 (Fed. Cir. 2011).

“stream communication” connection

- Plain claim language of dependent claim 15 pertains to “instantiation” term in the independent claim
- Petitioner and its expert even argued “instantiation” means to “create, such as by configuring, a particular structure.”

4. “instantiate”

In the context of the 324 Patent, the ordinary meaning to a Skilled Artisan of

“*instantiate*” is create, such as by configuring, a particular structure. EX1003, ¶

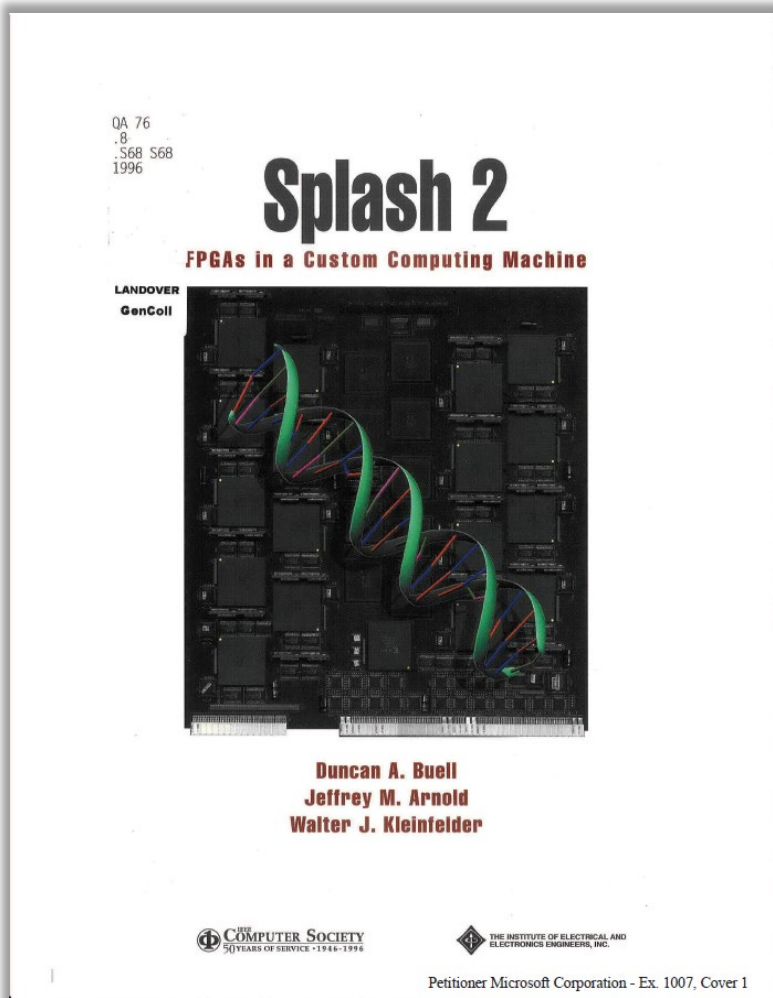
85.

- Petition at 16.
- Petitioner ignores the plain claim language that states the “stream communication” type of connection in Claim 15 further specifies the structure that is instantiated in Claim 1.

“stream communication” connection

- Petitioner’s proposed construction for stream communication is inconsistent with the foregoing.
- Instead, Petitioner’s construction abandons any structure or configuration thereof. This would render the dependent claim nonsensical and illogical by divorcing it from any of the requirements of the “instantiating” limitation.
 - Petitioner’s construction offers nothing clarifying about what an instantiation comprising stream communication would constitute.
 - Petitioner’s construction does nothing more than recite a requirement of sending data to any systolic (or even data processing) system generally—namely that it receives a sequence of data via some communication.

“stream communication” connection



- Neither Splash2 nor any of the other prior art disclose stream communication as properly construed, and Microsoft does not dispute this. See Reply, 37-38.
- The Petition asserts only that, based on Petitioner’s incorrect construction of stream communication as a sequence of data, this limitation is met by Splash2.

“stream communication” connection

- Splash2 does not contain any disclosure of
 - a queue between processing elements. EX2111¶¶188-193. Petitioner does not identify any in the Petition and Petitioner’s expert likewise is silent on any such disclosure in Splash2.
 - the signaling for the processing elements to interact with the queue so as to store/fetch, which a POSITA would know is necessary to do any sort of data communication involving a queue in a stream communication. EX2111¶¶180-193.
 - a busy/ready signal between processing elements to regulate the flow of data into and out of the queue. EX2111¶¶180-193.
- Instead, Petitioner and its expert argue Splash2 shows a “direct” connection between processing elements with nothing in between. 601 Petition at 12, 37, 46; EX 2064 at 85:14-86:12.

“stream communication” connection

- No structure is identified for Splash2 or any of the other prior art references, in accordance with a POSITA’s understanding that claim15 requires instantiating reconfigurable resources for “stream communication.” See EX2111 ¶¶150-154, 180-193.
- Nor is there any argument or evidence to show a combination of teachings from Splash2 and any of the other prior art to render obvious the limitation of instantiating some structure using the reconfigurable resources for “stream communication.”

“stream communication” connection

- Petitioner’s entire interpretation of Splash2 is that there is nothing in between the processing elements in order to be “seamless,” which Petitioner’s expert confirmed in deposition. Petition at 12, 37, 46; EX2064 at 85:14-88:10.

Page 85

1 STONE, Ph.D.
2 Oh, there it is. Okay.
3 So this concept of systolic, what's
4 your opinion of what that means?
5 A. My opinion of what it means is in
6 the report. It means it's the characteristic
7 of rhythmically computing and passing data
8 directly between processing elements. And then
9 I quote: "Without a program counter or clock
10 that drives a movement of data," and also
11 operating in a manner that is, "transport
12 triggered, i.e., by the arrival of a data
13 object."
14 Q. Okay. And you mention the word
15 "directly," it was passing data directly
16 between processing elements. What does that
17 phrase mean to you or what's the context?
18 What are you trying to describe
19 there?
20 A. That the data goes from first to the
21 second without going to something intervening.
22 It directly go -- is connected immediately.
23 Indirectly we -- you go through one or more
24 intervening places to get there.
25 Q. Okay. So would memory, if the data

Page 86

1 STONE, Ph.D.
2 was going from one processing element to memory
3 and then back to a processing element, is that
4 something you would consider as an intervening
5 thing?
6 A. Well, that would not be a direct
7 connection of the output of the cell to the
8 next cell. It says, "Between processing
9 elements you're directly connected." If you're
10 saying you have a processing element outputting
11 to memory and then coming back to another
12 processing element, that would not be direct.
13 Q. Okay. Are there any other examples
14 of intervening structures or circuits that
15 would violate this direct connection?
16 A. I -- I think you're opening a
17 universe. I'm not going to answer that because
18 I'd like to -- let's get specific things.
19 Q. Well, how about a -- a register?
20 Would that be an intervening structure?
21 A. I -- I'm puzzled because that --
22 that register would be within -- within the
23 processing element in my mind.
24 Q. Okay.
25 A. If it's within the processing

No Motivation to Combine

- “The presence or absence of a motivation to combine references in an obviousness determination is a pure question of fact.” *Intelligent Bio-Sys.*, 821 F.3d at 1366 (quoting *Par Pharm.*, 773 F.3d at 1196). Where a combination of prior art references changes the basic principles of operation of the prior art or renders the prior art inoperable for its intended purpose, there is no motivation to combine.
- MPEP 2143.01 (“If [the] proposed modification would render the prior art invention being modified unsatisfactory for its intended purpose, then there is no suggestion or motivation to make the proposed modification.”) (citing *In re Gordon*, 733 F.2d 900 (Fed. Cir. 1984));
- *Plas-Pak Indus. v. Sulzer Mixpac AG*, 600 Fed.Appx. 755, 759-60 (Fed. Cir. 2015) (“How well a combination is expected to work is certainly a legitimate consideration in an obviousness inquiry.”).

Dr. Stone is Not Reliable

Indeed, Dr. Stone dedicates an entire chapter in a textbook he authored in 1987, and subsequent editions in 1990 and 1993, about minimizing overhead in the design of such high-performance systems. EX2137, 281-330; EX2118. And, he notes if the overhead is not kept below a certain percentage of execution time, such parallel execution is not beneficial. EX2137, 289,329. Dr. Stone also readily admits that in evaluating the use of “exotic technology,” considerations of more reasonable technologies should be considered *rather than arbitrarily committing to a specific use of technology, such as his targeted focus solely on FPGAs here.* EX2137, 301.

And, Dr. Stone fully appreciates that a computer architect (and, in this case, his own POSITA) would take into consideration bottlenecks and efficiency in design. EX2137, 281-282, 288-289, 291, 299. Yet, in evaluating the prior art, the patent teachings, or the rationale underpinning to combine, Dr. Stone is silent about his own prior methodology.

Dr. Stone is Not Reliable

DirectStream's patents. Its expert's opinions now are not credible in light of his own prior published works, where such exotic and successful use of reconfigurable parts at the time of the invention were unknown; his selective discussion of the then state of the art; and his selective evaluation of the factual record (albeit by Petitioner's cherry-picking decision to show "facts" or "opinions" it only deemed relevant). EX2048 38:23-39:5, 52:2-55:3, 89:5-23, 142:22-144:10, 145:9-146:25, 161:13-162:25, 168:9-169:4, 179:6-13, 187:9-188:11, 203:14-207:13, 210:19-25; EX2058 20:4-21:14, 33:5-13, 35:17-36:20, 39:11-21, 66:17-67:2, 80:10-20, 207:8-209:21.

Dr. Stone is Not Reliable

- For example, in deposition, Dr. Stone testified he assumed his use of prior art in various combinations would be enabling without explaining how a POSITA would actually make that combination:

Page 54

1 HAROLD S. STONE, Ph.D.
2 combinations. I'm more thinking and asking
3 generally across your declaration, this issue
4 of whether a person of ordinary skill in the
5 art at the time of the invention could practice
6 whatever the disclosure is in the prior art
7 reference, how did you go about making that
8 assessment?
9 A. Let's have an example of prior art
10 reference 1 and 2, and I would argue -- look at
11 it carefully, and I would decide whether one of
12 skill in the art would understand that they
13 could be combined technically, that there's
14 motivation to combine them. If you combined
15 them, that you would get the results and the
16 results are relevant to the analysis of
17 invalidity.
18 As far as enabling, my argument
19 would be that you're enabled to put them
20 together. The individual prior art I would
21 assume are enabled because they're patents.
22 I -- I don't -- I did not specifically
23 investigate each of the patents to see if they
24 were enabled to do what they've claimed to do.
25 So my assumption is they could do what they

Page 55

1 HAROLD S. STONE, Ph.D.
2 could do and that a person of ordinary skill in
3 the art would -- was enabled to combine them.
4 Q. Okay. So looking at the phrase that
5 you just used, "the person of ordinary skill in
6 the art," and I want to zero in on this idea of
7 a person of ordinary skill in the art at the
8 time of invention.
9 Do you understand what that phrase
10 means?
11 A. I do.
12 Q. And would you agree with me that
13 that person of ordinary skill in the art --
14 sorry. Let me -- let me rephrase that.
15 Would you agree with me that looking
16 at a prior art reference from the perspective
17 of a person of ordinary skill in the art at the
18 time of the invention precludes using what we
19 would call hindsight bias?
20 A. I understand that.
21 Q. So you understand what hindsight
22 bias is?
23 A. I believe I do, but again, I'm not
24 a -- a legal expert. I'm not -- it's just from
25 experience and testimony.

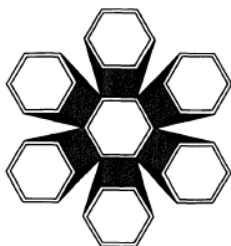
- EX2066 at 54:2-55:3 (emphasis added)

Dr. Stone is Not Reliable

- Petitioner's expert also testified he merely assumed all the benefits and ignored any of the drawbacks. EX2066:
 - 65:17-68:22 (state of the art would include what you cannot do also but Stone doesn't consider the disadvantages in his opinion, only advantages)
 - 142:22-144:10 (Stone did not rely on his own book and didn't consider disadvantages identified in his book for multiprocessor systems)
 - 145:9-146:25 (didn't consider disadvantages when considering prior art combinations)
 - 161:13-162:25 (Stone believes legal standard does not teach him to take into consideration whether a particular combination is even a good combination in terms of the system that results)
 - 187:9-188:11 (Stone didn't investigate potential drawbacks of FGPA's in 2001)
 - 203:14-207:13 (Stone only used perceived advantages to piece together combinations without considering whether the combinations were workable)

Dr. Stone is Not Reliable

High-Performance Computer Architecture



Harold S. Stone
IBM Watson Research Center
and
Courant Institute
New York University

◆ **Addison-Wesley Publishing**

Reading, Massachusetts
Menlo Park, California
Wokingham, England
Sydney • Singapore • Toronto
Bogotá • Santiago • San

Moreover, in considering any proposed computer architecture implementation, a POSITA would have known of the need to analyze the costs and benefits associated with the various overhead from introducing “exotic designs,” which even Petitioner’s own expert admitted in writings around the time of the invention. EX2095¶¶238, 244. This includes benchmarking and bottleneck analysis to measure the power, performance, cost, etc. associated with a system architecture and available solutions for each of them, and may even involve trial and error.

EX2095¶¶241-242; *see e.g.*, EX2138 (an example of such an analysis that would be known to a POSITA). Petitioner’s expert agreed in deposition such an analysis is proper, but he failed to perform one in any of his analysis in this matter.

EX2095¶¶239, 243-247.

PATENT OWNER DIRECTSTREAM, LLC
EX. 2052, p. 3

Dr. Stone is Not Reliable

As discussed by Dr. Homayoun, a proper POSITA would have considered all of the state of the art in the design of computer architecture, including for example, the issues of reconfigurable programming, processor speed, FPGA speed, benchmarking, bottlenecking, and cost/benefit analysis of overhead introduction as applied to HPC applications. EX2095¶¶99-106, 134-148; EX2136, 41, 45, 67-74, 363-387. As shown above, even Dr. Stone's own prior written admissions concur that POSITA would consider these issues. *See* EX2052; EX2137; EX2118. But Petitioner ignored this basic analysis any POSITA should have undertaken. The fundamental flaw of Petitioner's arguments is the hindsight bias, as discussed above, to focus on FPGAs as the solution to problems in high performance computing. EX2095¶¶143-148.

Dr. Stone is Not Reliable

- Even assuming an algorithm could be moved from a multiprocessor to an FPGA environment, there are additional design considerations that must be evaluated to determine if any FPGA implementation will be successful. EX2111¶¶242-252.
- For example, chapter 2 of the book Field-Programmable Gate Array Technology by Dr. Trimberger also provides additional design considerations that must be taken into account when evaluating whether it is possible to implement an algorithm on an FPGA or array of FPGAs:

Dr. Stone is Not Reliable

- EX2078 at 29.
 - Capacity Estimation
 - FPGAs have three kinds of resources: logic, I/O and routing. To determine if a design fits into a particular FPGA, the design must fit within all three resource limits. The difficulty of this estimation is a function of the architecture and of the software used for mapping the logic into the FPGA. FPGA logic and interconnect capacity are difficult to estimate. Traditional measures of gate count and product terms are not accurate estimates of lookup-table capacity. Two designs that appear to be of equal size in terms of FPGA gate count or number of PLD product terms may use CLBs with different efficiency, requiring very different numbers of CLBs. Logic optimization algorithms may also significantly change the size and performance of the design. Complex blocks implement complex functions efficiently, but when the function to be implemented does not fit into the block efficiently, some fraction of the block is unusable, and is wasted.

Dr. Stone is Not Reliable

- EX2078 at 29.
 - The wasted fractions of blocks cause a gap between the peak capacity of the FPGA and the capacity in a given application. An accurate capacity and performance estimate requires that the design be mapped into the FPGA. Fortunately, fast mapping heuristics can give a good estimate of logic capacity. Routing requirements are more difficult to estimate. The problem of statistical wirability estimation has been addressed by Heller [1978], Donath [1979] and ElGamal [1981], but the techniques and results are not accurate enough for capacity estimation. MPGA designs address this problem by providing significantly more interconnect than is needed by most designs. This solution is impractical in FPGAs because unused FPGA interconnect degrades performance and density too severely. FPGA design systems include high-speed placement and routing for routability estimation and timing-driven routing to meet delay requirements.

Dr. Stone is Not Reliable

- EX2078 at 66-67.
 - FPGA interconnect is comparatively expensive, both in terms of delay and area. An architecture that includes more long-distance connections would have faster interconnect. but the resulting chips might require more area for interconnect. reducing their logic capacity. Architectures with minimal interconnect resources will appear denser, but might be difficult to route. Architectures must address both integrated circuit and software goals. The true capacity and speed of an FPGA is measured by the ability of design automation software to exploit the architecture. FPGA architectures and software must be developed simultaneously.

Dr. Stone is Not Reliable

- EX2078 at 66-67.
 - Software
 - The CAE industry has focused on the MPGA problem and has adopted a gate-like implementation model based on MPGA features. Many of the current software issues with FPGAs are a result of their non-gate-like implementation structure. This disagreement is most evident in the schematic entry library, which is a collection of gate-level primitives. The netlist generated from a schematic preserves the gate-like structure. The non-gate-like FPGA structure requires a partitioning step before the placement and routing process. Related problems in design automation have been addressed either as placement, considering only the physical constraints; or as technology mapping, considering only the logical constraints. Both sets of constraints must be solved simultaneously in order to produce implementations that are simultaneously dense and fast. The partitioning problem is aggravated by the use of logic optimization algorithms originally designed for gate-like implementations.

Dr. Stone is Not Reliable

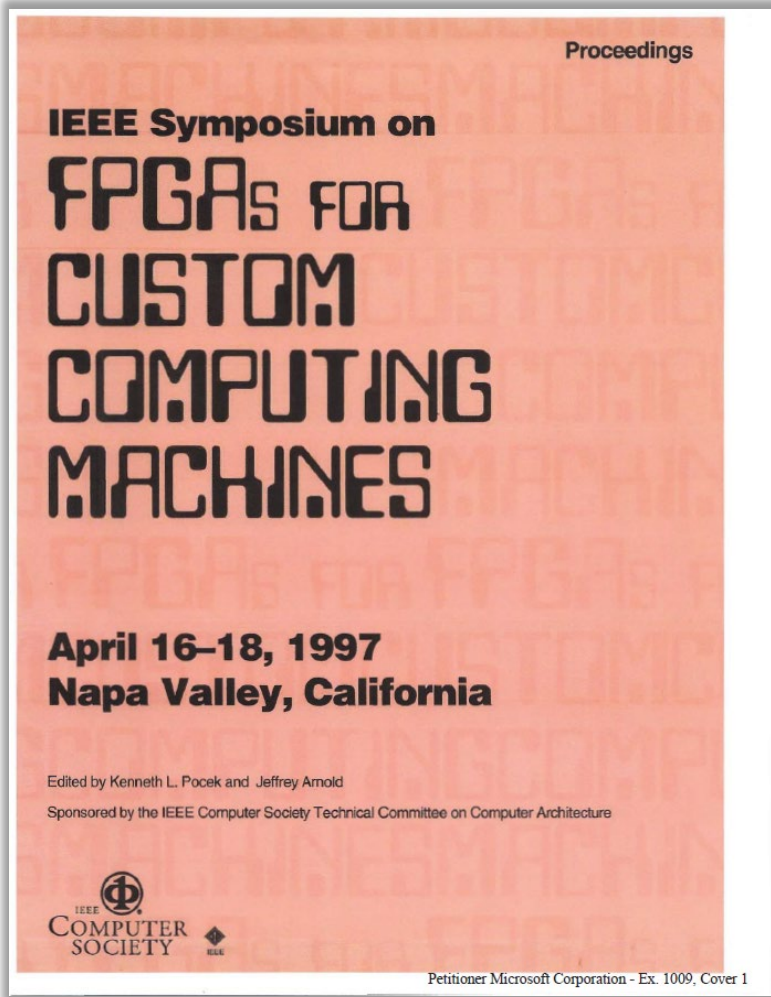
- EX2078 at 66-67.
 - They often produce results that reduce speed and density rather than improve them. The reasons are varied, but traditional algorithms tend to factor logic aggressively, making more small gates; they ignore the ability of lookup tables to subsume larger amounts of logic. They also ignore routability considerations, which are of vital importance to FPGAs. New optimization algorithms are needed for lookup-table based FPGA architectures.
 - High-level synthesis and logic synthesis systems must target the high-level architectural features of FPGAs to gain the performance and density advantages they provide. The Library of Parameterized Macros (LPM) [Holley 1991] is an industry sponsored standardization effort to develop an intermediate form that includes these high-level constructs. It may provide the appropriate interface between high level synthesis systems and systems-oriented FPGAs. Placement and routing of FPGAs provides new challenges.

Dr. Stone is Not Reliable

- EX2078 at 66-67.
 - The relatively slow FPGA interconnect structure demands true timing-driven placement and routing algorithms. Although these algorithms have been proposed for MPGA design automation, their usefulness for MPGA designs has not been great, and their adoption for FPGAs seems to be happening more quickly.
 - Partitioning in Space and Time
 - Because of the limited capacity of FPGAs, and their applicability to prototyping, FPGAs have re-kindled interest in multi-chip partitioning. There are several important problems that must be addressed, including FPGA resource estimation (logic, I/O and routing), timing and partitioning into dissimilar parts. A farther-reaching problem is the issue of partitioning a design in time: identifying parts of a design that can be time-shared onto the FPGA, and generating separate FPGA configurations for them. At present, not only are there no algorithms, but the current design representations appear to be lacking in essential timing information. An elegant solution to this problem will allow true time-shared hardware and usher in a new era in hardware implementation.

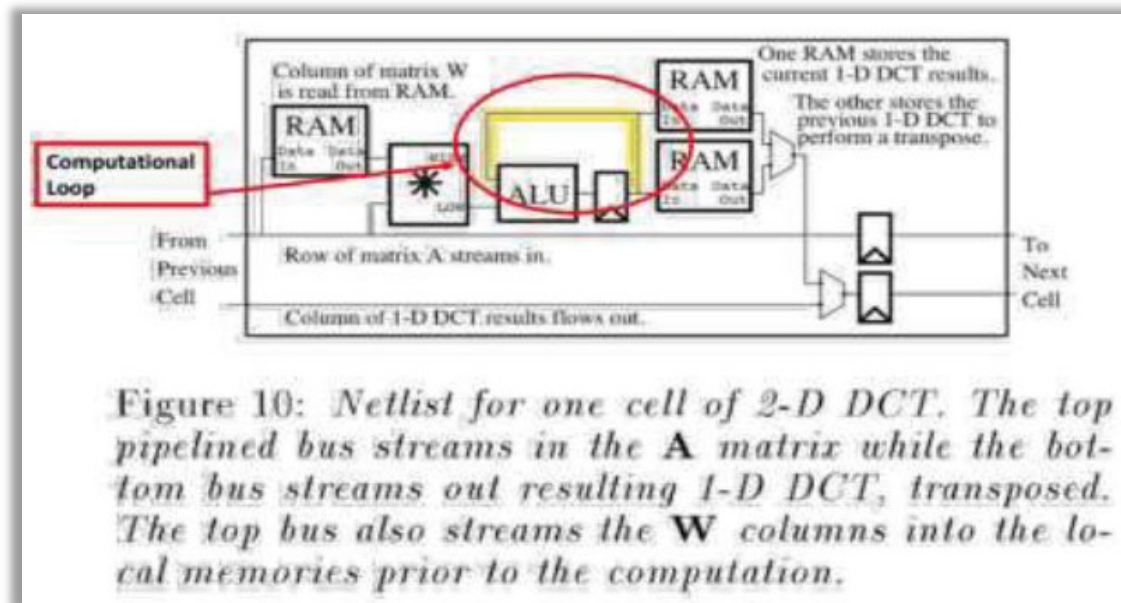
Other prior art - RaPiD

- EX1009 - RaPiD prior art
- Not taught in RaPiD



Other prior art - RaPiD

- The Petition contains only a conclusory allegation that RaPiD discloses a “computational loop” based on an annotated Figure 10. 601 Petition at 63.
- The only evidence that Petitioner and its expert point to is the below annotated (but it is unclear who annotated it) Figure 10 from RaPiD.



Other prior art - RaPiD

- A forwarding path is used to “support back to back execution of operations without stall, by forwarding (or bypassing) the output of an ALU to an input of the same or other ALU ... for back to back operation without adding any stall.” EX2111¶236; EX2029¶45;

forwarding Also called **bypassing**. A method of resolving a data hazard by retrieving the missing data element from internal buffers rather than waiting for it to arrive from programmer-visible registers or memory.

really exist when doing laundry). For example, suppose we have an add instruction followed immediately by a subtract instruction that uses the sum (\$s0):

```
add    $s0, $t0, $t1
sub    $t2, $s0, $t3
```

Without intervention, a data hazard could severely stall the pipeline. The add instruction doesn't write its result until the fifth stage, meaning that we would have to waste three clock cycles in the pipeline.

Although we could try to rely on compilers to remove all such hazards, the results would not be satisfactory. These dependences happen just too often and the delay is just too long to expect the compiler to rescue us from this dilemma.

The primary solution is based on the observation that we don't need to wait for the instruction to complete before trying to resolve the data hazard. For the code sequence above, as soon as the ALU creates the sum for the add, we can supply it as an input for the subtract. Adding extra hardware to retrieve the missing item early from the internal resources is called **forwarding** or **bypassing**.

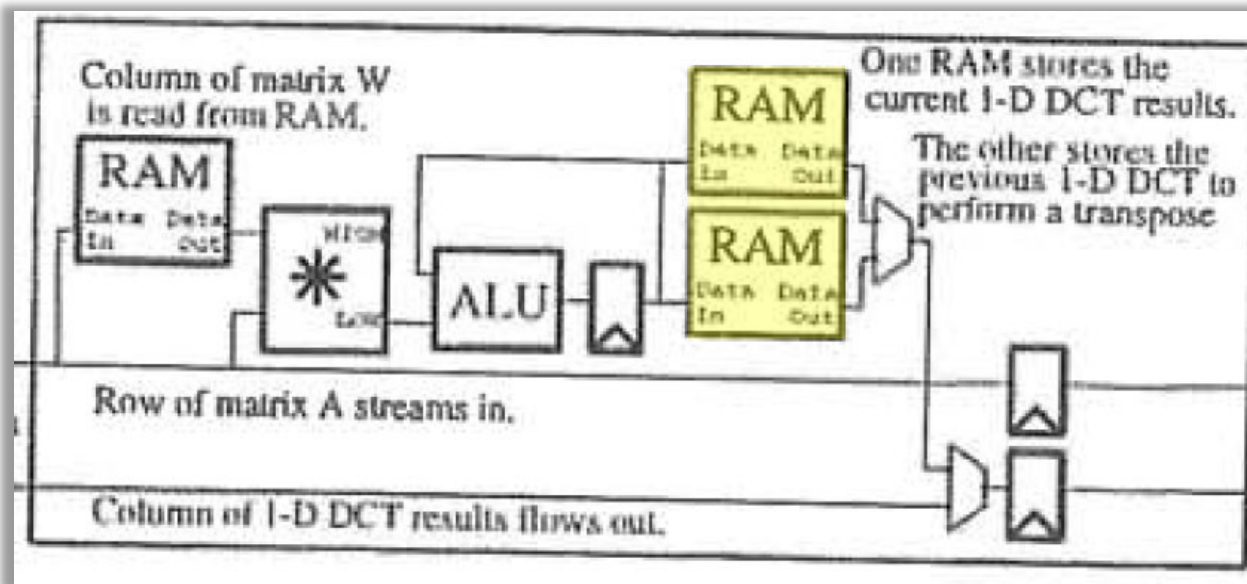
- EX2043 at 301; EX2044 at 6. And it is one of the basic structures in computer microarchitecture. EX2111¶¶237; EX2044 at 2, 6-7; EX2029¶46.

Other prior art - RaPiD

- Petitioner does not explain why it believes the yellow path in Figure 10 is a “computational loop.”
- Petitioner’s expert concludes that it is because “the output [of] the ALU is looped back to the ALU input.” EX1003¶357.
 - That is the same as the definition of a forwarding path
- Petitioner’s expert even conceded Fig. 10 depicts only a forwarding path or a bypass path. EX2111¶238
- EX2064 at 201:21-202:1:
 - 24 Q. Does the Figure 10 show a bypass
 - 25 path or a forwarding path?
 - 2 A. Yes.

Other prior art - RaPiD

- The Petition asserts only a conclusory statement that “DCT data is passed ‘seamlessly’ in that such data is communicated directly from one Processing Element to the next.” Petition at 64. Petitioner’s expert that repeats the Petition’s conclusory statements verbatim. EX1003¶360.
- Figure 10 in RaPiD clearly shows storage of results in memory (RAM) before being passed onto the next cell:



- EX1009 at 111; 601 Petition at 63.

Other prior art - RaPiD

- RaPiD clearly discloses the exact structure that Petitioner's expert testified would not be seamless.

Page 85

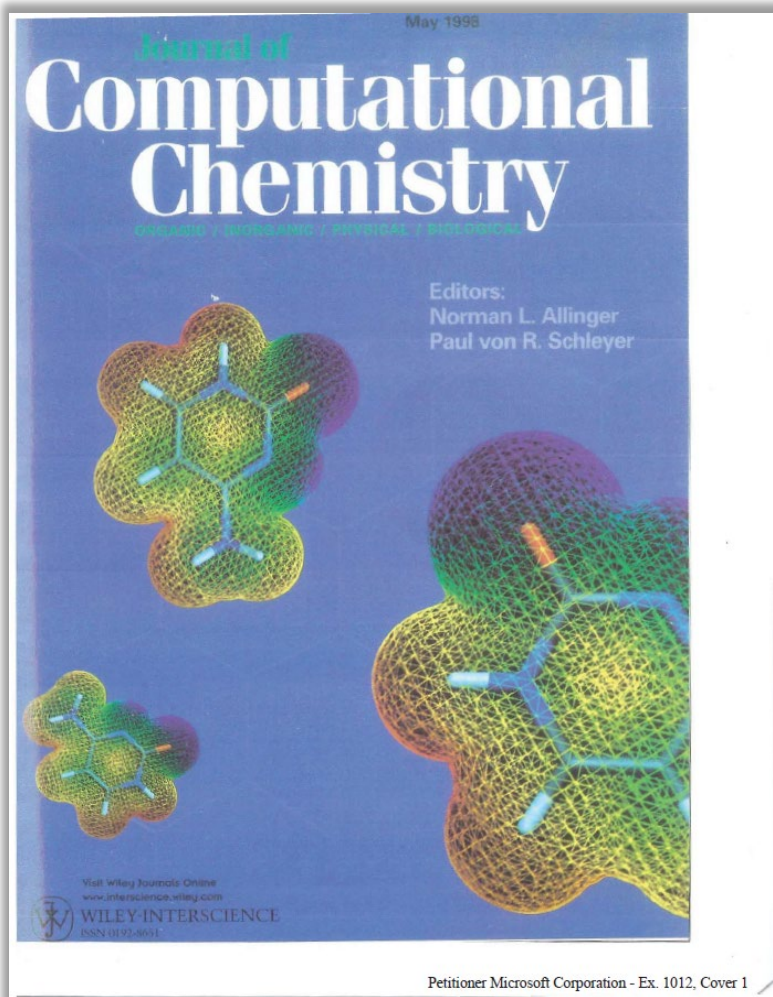
1 STONE, Ph.D.
2 Oh, there it is. Okay.
3 So this concept of systolic, what's
4 your opinion of what that means?
5 A. My opinion of what it means is in
6 the report. It means it's the characteristic
7 of rhythmically computing and passing data
8 directly between processing elements. And then
9 I quote: "Without a program counter or clock
10 that drives a movement of data," and also
11 operating in a manner that is, "transport
12 triggered, i.e., by the arrival of a data
13 object."
14 Q. Okay. And you mention the word
15 "directly," it was passing data directly
16 between processing elements. What does that
17 phrase mean to you or what's the context?
18 What are you trying to describe
19 there?
20 A. That the data goes from first to the
21 second without going to something intervening.
22 It directly go -- is connected immediately.
23 Indirectly we -- you go through one or more
24 intervening places to get there.
25 Q. Okay. So would memory, if the data

Page 86

1 STONE, Ph.D.
2 was going from one processing element to memory
3 and then back to a processing element, is that
4 something you would consider as an intervening
5 thing?
6 A. Well, that would not be a direct
7 connection of the output of the cell to the
8 next cell. It says, "Between processing
9 elements you're directly connected." If you're
10 saying you have a processing element outputting
11 to memory and then coming back to another
12 processing element, that would not be direct.
13 Q. Okay. Are there any other examples
14 of intervening structures or circuits that
15 would violate this direct connection?
16 A. I -- I think you're opening a
17 universe. I'm not going to answer that because
18 I'd like to -- let's get specific things.
19 Q. Well, how about a -- a register?
20 Would that be an intervening structure?
21 A. I -- I'm puzzled because that --
22 that register would be within -- within the
23 processing element in my mind.
24 Q. Okay.
25 A. If it's within the processing

- EX2064 at 85:14-86:18.

Other prior art - Roccatano



- EX1012 Roccatano prior art
- Not taught in Roccatano

Other prior art - Roccatano

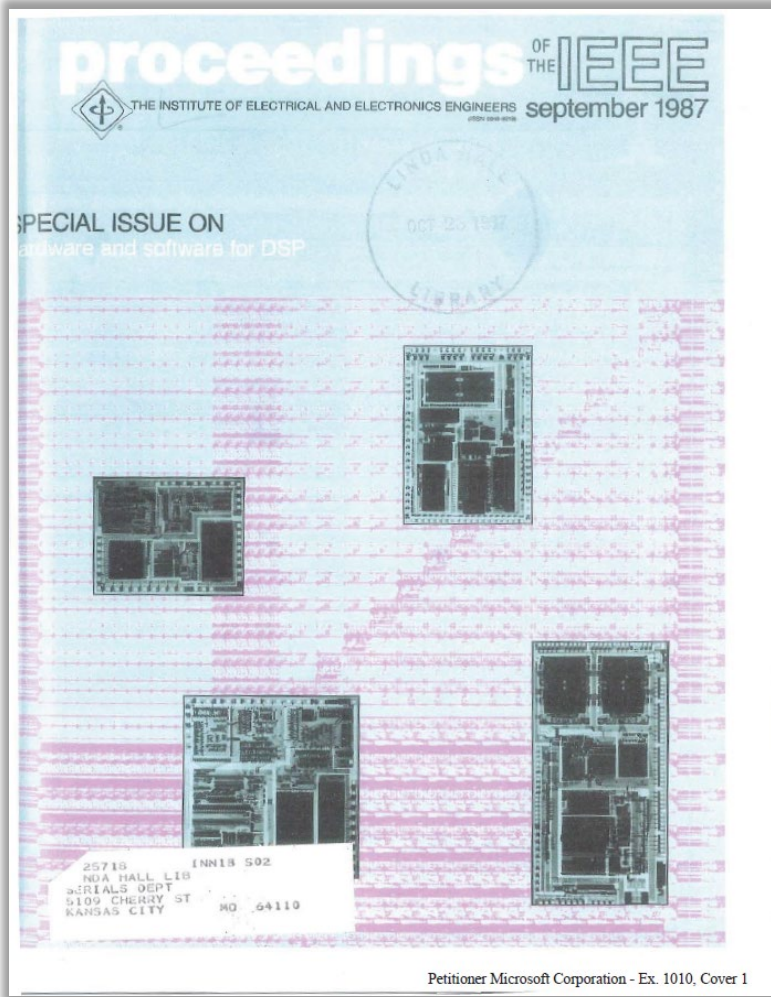
- Petitioner again argues that computational loops are disclosed in Roccatano, which “accumulates the interactions of a resident atom and a transient atom for a collection of resident-transient atom pairs [and t]he accumulation occurs for a number of steps equal to $P/2$, where P is the number of processors... operating on a different resident atomic data...” 603 Petition at 76; EX1003¶¶305.
- This is the same argument Petitioner advances for Splash2, where the “looping” is just to execute the code once for each piece of data. EX2111¶¶231-233.
- There is no disclosure of looping or repeating of a computation multiple times for each data until a condition is met or a number of repetitions has been satisfied, as required by the Board’s claim construction and the ’324 Patent. EX2111¶¶231-233.

Other prior art - Roccatano

- Roccatano does not disclose “seamless” because its teachings require multiple processors with the exact inherent boundaries from chip-to-chip communication that the '324 Patent sought to address. See supra §§II.C and V.A.1. Roccatano clearly discloses “8 to 2048 processors... arranged in a three-dimensional (3D) cubic mesh.” EX1012 at 686.
- This is no different than the prior art the applicant distinguished during prosecution and therefore cannot disclose seamlessly passing data between computational loops. See EX1002 at 117-118, 147-148, 174-175, 224-225.
- Roccatano also discloses each processor having local memory of up to 4 megabytes, similar to the local memory in Splash2. Even if Roccatano did not require multiple processors, it still would be ambiguous whether the processors stored intermediate results in this available local memory, which would have been expected in the prior art at the time of the invention to smooth over at least the timing problems inherent in systolic systems. See EX2111 ¶¶210-220.

Other prior art - Gaudiot

- EX1010 Gaudiot prior art
- Not taught in Gaudiot



Other prior art - Gaudiot

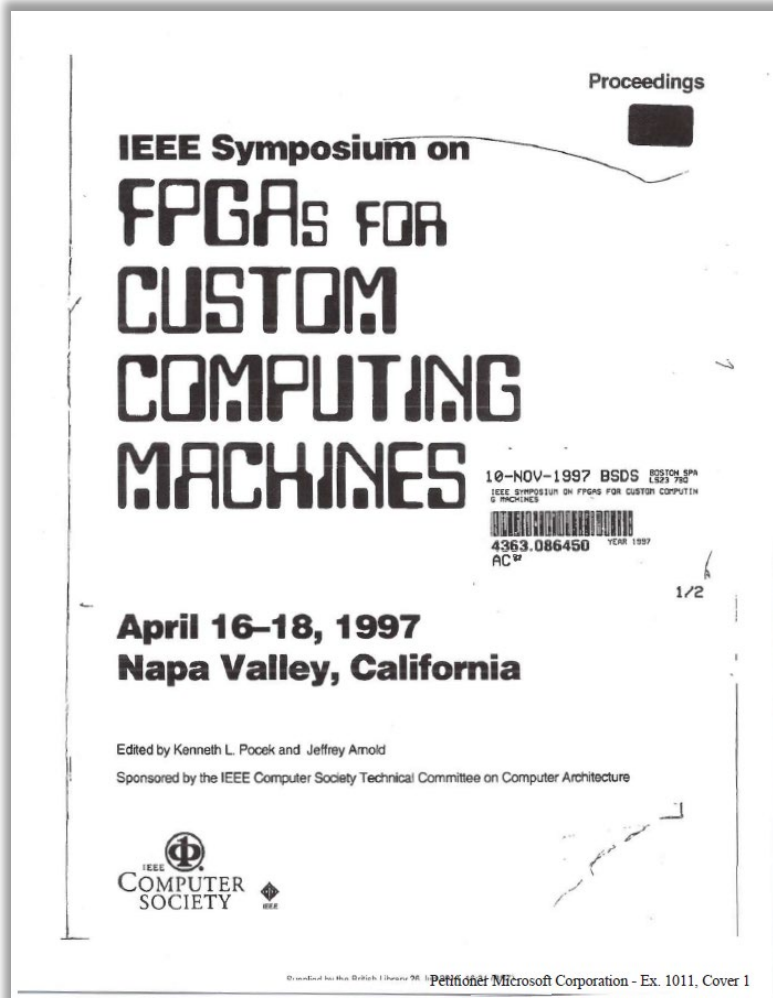
- Neither the Petition nor the report of Petitioner's expert contain any argument or evidence that Gaudiot discloses two computational loops. 601 Petition at 52-53; EX1003¶¶195-201.
- Neither the Petition nor the report of Petitioner's expert contain any argument or evidence that Gaudiot discloses passing computed data seamlessly between two computational loops. 601 Petition at 52-53; EX1003¶¶195-201.
- Gaudiot discusses the broad data processing concepts with respect to systems with multiple processors, not systems having multiple processing elements on a single chip as claimed in the '324 Patent. 601 Petition at 52-53 (Petitioner admits "Gaudiot discloses a multiprocessor technique..."); see also EX1003¶195.

Other prior art - Gaudiot

- Petitioner admits “Gaudiot discloses a multiprocessor technique...,” not systems having multiple processing elements on a single chip as claimed in the '324 Patent. 601 Petition at 52-53; see also EX1003¶195.
- Petitioner and its expert try to combine the teachings of Gaudiot with Splash2 but do not discuss any of the above considerations that would be relevant to a POSITA. In fact, Petitioner’s expert even failed to follow his own rubric for analyzing whether a POSITA would be motivated (or even consider it feasible) to modify any of the prior art as he proposes in his report to meet the claim limitations of the '324 Patent.

Other prior art - ChunkySLD

- EX1011 ChunkySLD prior art
- Not taught in ChunkySLD



Other prior art - ChunkySLD

- Petitioner argues that computational loops are disclosed in ChunkySLD, which “accumulates the partial products into a dot product of two column vectors [and t]he accumulation occurs for a number of time slots equal to the length of a template column.” 602 Petition at 72; EX1003¶¶405.
- This is the same argument Petitioner advances for Splash2, where the “looping” is just to execute the code once for each piece of data. EX2111¶¶244-226.
- There is no disclosure of looping or repeating of a computation multiple times for each data until a condition is met or a number of repetitions has been satisfied, as required by the Board’s claim construction and the ’324 Patent. EX2111¶¶224-226.

Other prior art - ChunkySLD

- ChunkySLD discloses that the summation result from a processing element must be held in storage for at least one extra time step before being passed to the next processing element. EX1011 at Fig. 5; EX2111 ¶¶210-219.

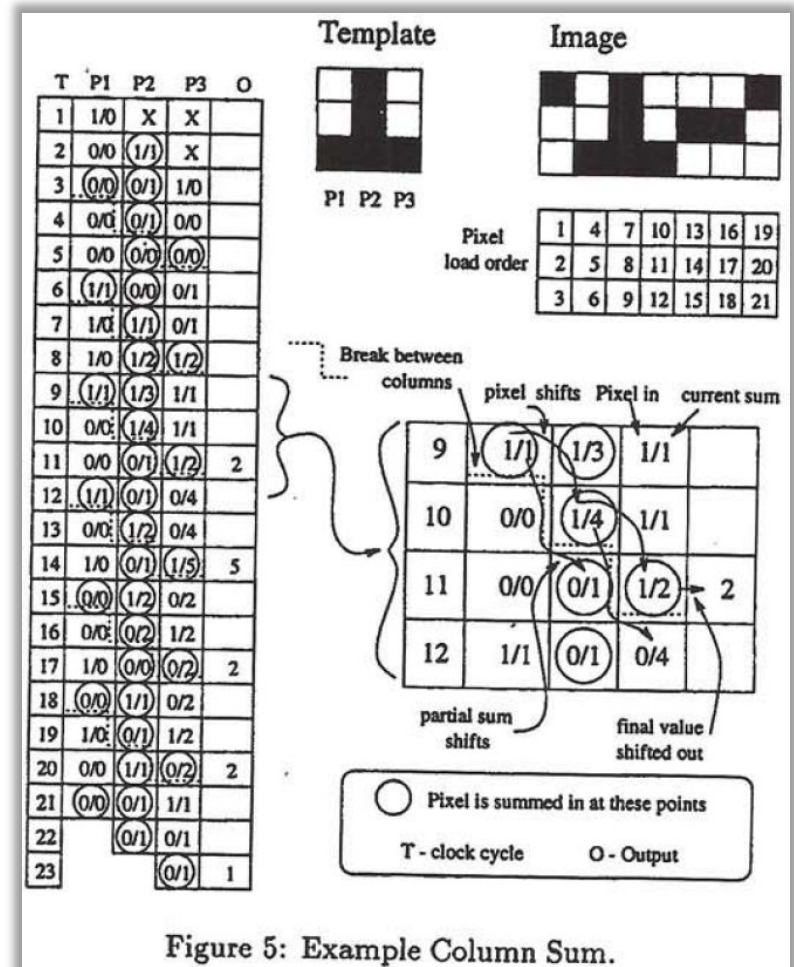


Figure 5: Example Column Sum.

Other prior art - ChunkySLD

- Petitioner's expert also admits ChunkySLD requires this delay in transferring between processing elements. EX1003 ¶¶389-390;

389. In the first three time slots, cell P1 calculates the dot product of the column P1 of the template with the first column of the image. This is the dot product of (0,0,1) with (1,0,0).

390. At time 4 the saved partial product from time 3 is passed from cell P1 to cell P2. In the meantime, cell P2 operates similarly to cell P1, but is delayed by one cycle and uses column P2 of the template instead of column P1. Thus, at times 2 through 4, cell P2 calculates the dot product of column 1 of the image (1,0,0)

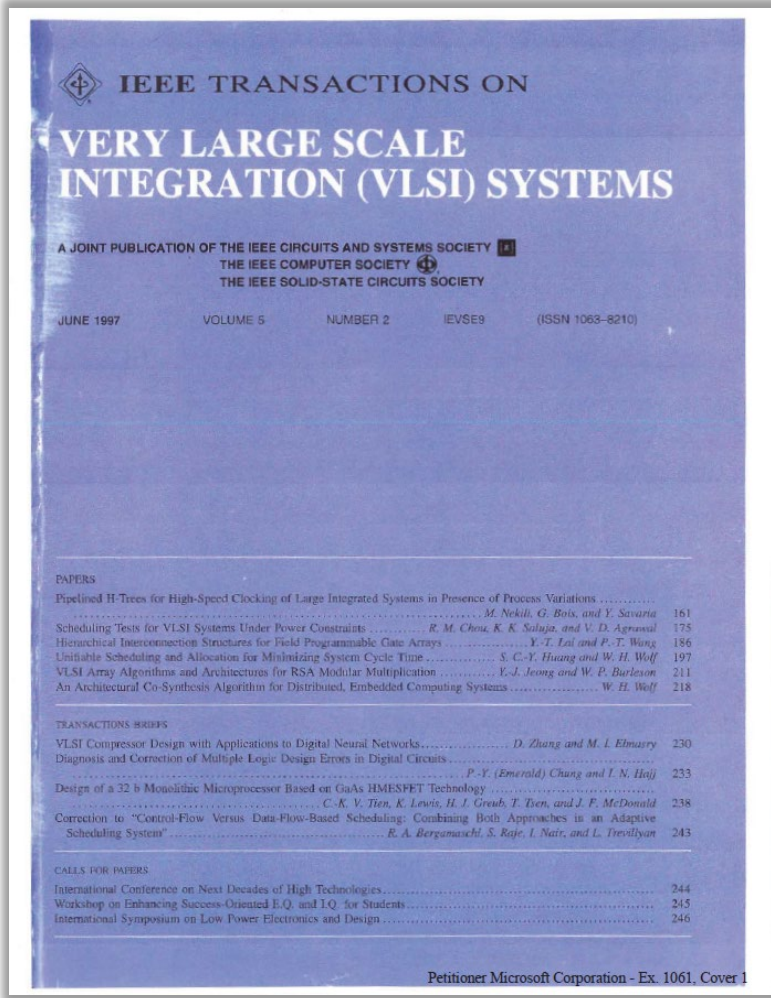
- EX2064 at 194:24-195:4
24 Q. Okay. Okay. Okay. So that I -- I
25 am reading that right, that the
26 2 time step to -- to I guess move across that
27 3 break between the columns?
28 4 A. Right.

Other prior art - ChunkySLD

- ChunkySLD is simply an algorithm deployed on Splash2, and therefore the same ambiguity present in Splash2 is also present for other deployments of the same platform. EX1011 at 195; see EX2111 ¶¶210-219.
- At best, ChunkySLD is ambiguous where this result is stored while waiting multiple time steps to be passed to the next processing element.
- Thus, similarly with Splash2, it is equally, if not more, plausible to interpret ChunkySLD as disclosing the need to store results in memory, for example in the local memory attached to the FPGA, to account for the timing issues above. See EX2111 ¶¶210-220.

Other prior art - Jeong

- EX1061 Jeong prior art
- Not taught in Jeong



Other prior art - Jeong

- Petitioner argues that computational loops are disclosed in Jeong to perform “modular multiplication.” 601 Petition at 72-73; EX1003 ¶¶451-452.
- Jeong describes this modular multiplication algorithm as an iterative procedure which performs a single summation from $i=0$ to $n-1$ (the equivalent number of iterations as going from 1 to n). EX1061 at 212; EX2111 ¶¶227-230.

Now we describe the general modular multiplication algorithm using the *modulo during multiplication* approach. Given any two n -bit integers, A and B , and the n -bit modulus C , where $(C > A, B)$, the modular multiplication can be described by an iterative procedure using *Horner's rule*

$$\begin{aligned} AB \bmod C &= A \cdot \sum_{i=0}^{n-1} b_i 2^i \bmod C \\ &= ((\cdots (b_{n-1} A)2 + b_{n-2} A)2 \\ &\quad + \cdots + b_1 A)2 + b_0 A \bmod C. \end{aligned} \quad (2)$$

We can describe (2) in a recursive form as follows:

$$\begin{aligned} P_0 &= 0 \\ P_i &= 2P_{i-1} + b_{n-1} A \bmod C \end{aligned} \quad (3)$$

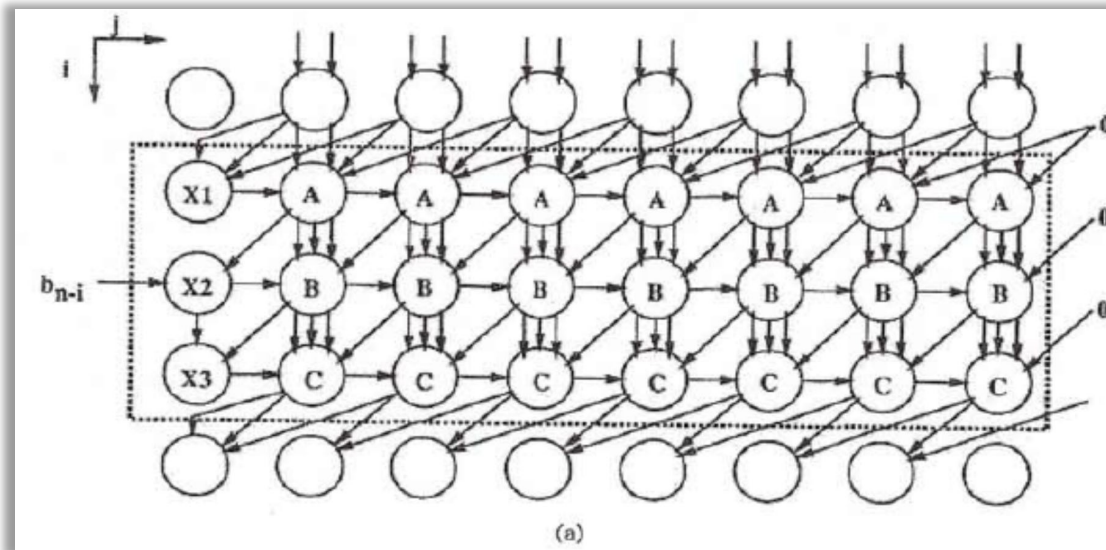
and P_n is the final result. Using (1) and (3), we will derive two different bit-level array structures.

Other prior art - Jeong

- Jeong even teaches to “precalculate the Kh’s instead of adding K multiple times” due to limitations on the number of allowable operands. EX1061 at 213; EX2111 ¶¶227-230.
- This is the same argument Petitioner advances for Splash2, where the “looping” is just to execute the code once for each piece of data. EX2111 ¶¶227-230.
- There is no disclosure of looping or repeating of a computation multiple times for each data until a condition is met or a number of repetitions has been satisfied, as required by the Board’s claim construction and the ’324 Patent. EX2111 ¶¶195, 230.

Other prior art - Jeong

- Splash2 is a linear system, requiring any implementations to be carefully planned to fit within the limited construction of Splash2. EX1011 at 194-197.
- Jeong discloses a non-linear system that cannot be deployed linearly, as shown below:



- EX1061 at 214.

Secondary Considerations

Obviousness

D. The Objective Indicia in this Case Indicate Nonobviousness

Evidence of “objective indicia of non-obviousness” is relevant to whether an invention is obvious over the prior art. *Graham*, 383 U.S. at 17-18; *see also, e.g., World Bottling Cap, LLC v. Crown Packaging Technology, Inc.*, IPR2015-01651, Paper 34 (P.T.A.B. Jan. 19, 2017). In fact, “[s]econdary considerations can be the most probative evidence of non-obviousness in the record, and enables the court to

121

avert the trap of hindsight.” *Crocs, Inc. v. International Trade Commission*, 598 F.3d 1294, 1310 (Fed. Cir. 2010) (emphasis added, internal quotes omitted); *see also Nike Inc. v. Adidas AG*, 812 F.2d 1326, 1339 (Fed. Cir. 2016) (objective indicia “may often be the most probative and cogent evidence in the record.”). Evidence of secondary considerations “is not just a cumulative or confirmatory part of the obviousness calculus but constitutes independent evidence of non-obviousness.” *Ortho-McNeil Pharm., Inc. v. Mylan Labs., Inc.*, 520 F.3d 1358, 1365 (Fed. Cir. 2008). Evidence of secondary considerations “must always when present be considered en route to a determination of obviousness...[and i]t is to be considered as part of all the evidence, not just when the decision maker remains in doubt after reviewing the art.” *In re Cyclobenzaprine*, 676 F.3d 1063, 1075-76 (Fed. Cir. 2012) (quoting *Stratoflex, Inc. v. Aeroquip Corp.*, 715 F.2d 1530, 1538-39 (Fed. Cir. 1983).

Various objective indicia of nonobviousness are supported and corroborated by evidence submitted by DirectStream. Once DirectStream is able to show a sufficient nexus between the indicia and the patented invention, the burden shifts back to Petitioner to prove that the indicia was due to other factors extraneous to the patented invention. *J.T. Eaton & Co., Inc. v. Atlantic Paste & Glue Co.*, 106 F.3d 1563, 1571 (Fed. Cir. 1997); *Apple Inc. v. Sightsound Techs., LLC*, CMB2013-00023 Paper 101 at 43 (P.T.A.B. Oct. 7, 2014).

C. The Objective Indicia in this Case Indicate Nonobviousness

Finally, DirectStream asserts that its alleged inventions are supported by objective indicia of nonobviousness, Response, 120-25, but nowhere does it tie these supposed objective considerations to the claims, let alone to the putatively

52

IPR2018-01601, -01602, -01603
(U.S. Patent No. 7,225,324)

novel elements of the claims, as required by the law. *See Polaris Indus., Inc. v. Arctic Cat, Inc.*, 882 F.3d 1056, 1072-73 (Fed. Cir. 2018).

Secondary Considerations

Long-Felt Need

Secondary Considerations

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEALS BOARD

MICROSOFT CORPORATION

Petitioner,

v.

DIRECTSTREAM, LLC,

Patent Owner.

IPR2018-01594 (Patent 6,434,687)
IPR2018-01599 (Patent 6,076,152)
IPR2018-01600 (Patent 6,247,110)
IPR2018-01601 (Patent 7,225,324)
IPR2018-01602 (Patent 7,225,324)
IPR2018-01603 (Patent 7,225,324 B2)
IPR2018-01604 (Patent 7,421,524 B2)
IPR2018-01605 (Patent 7,620,800 B2)
IPR2018-01606 (Patent 7,620,800 B2)
IPR2018-01607 (Patent 7,620,800 B2)

DECLARATION OF DR. TAREK EL-GEHAYOURY

PATENT OWNER DIRECTSTREAM, LLC
EX. 2164, p. 1

25. The amount of data movement required for single image processing was too large for use in an FPGA architecture that interfaced the FPGA through the PCI bus, especially for applications that do not require data reuse to amortize the cost of the transfers.

26. One of the breakthroughs that made supporting applications on FPGAs more feasible was SRC Computers' (SRC) development of interfacing the FPGA with the memory bus in the SRC 6E and later the Crossbar switch in the SRC 6. This allowed for faster data movement between the FPGA and the microprocessor.

Secondary Considerations

UNITED STATES PATENT AND TRADEM

BEFORE THE PATENT TRIAL AND APP

MICROSOFT CORPORATION

Petitioner,

v.

DIRECTSTREAM, LLC,
Patent Owner.

IPR2018-01594 (Patent 6,434,687)
IPR2018-01599 (Patent 6,076,152)
IPR2018-01600 (Patent 6,247,110)
IPR2018-01601 (Patent 7,225,324)
IPR2018-01602 (Patent 7,225,324)
IPR2018-01603 (Patent 7,225,324)
IPR2018-01604 (Patent 7,421,524)
IPR2018-01605 (Patent 7,620,800)
IPR2018-01606 (Patent 7,620,800)
IPR2018-01607 (Patent 7,620,800)

DECLARATION OF DR. TAREK EL-G

PATEN

28. By the early to mid-2000s, a few parallel computing systems with FPGAs supported by HPC vendors started to emerge, with the first ones being SRC and Starbridge. In 2002, and under a contract from the NSA LUCITE program, I led a team of multiple universities to study this emerging High-Performance Computing Technology (HPRC) with focus on SRC and Starbridge. The team included GW (Myself), GMU (Dr. Gaj) and USC (Dr. Buell). While the team had difficulties with the Starbridge product, the team was very satisfied with the initial tested machine from SRC, the SRC 6E. Eventually, Starbridge went out of business while SRC introduced its SRC 6 follow up. With the good experience from the SRC 6E in total productivity (performance plus user experience), the NSA and the team made the decision to purchase and continue further research investigations with the SRC 6. The

PATENT OWNER DIRECTSTREAM, LLC
EX. 2164, p. 10

Secondary Considerations

Teaching Away / Skepticism

Secondary Considerations

Page 1

1 STEPHEN M. TRIMBERGER, Ph.D.
2 UNITED STATES PATENT AND TRADEMARK OFFICE
3
4 BEFORE THE PATENT TRIAL AND APPEAL BOARD
5

6
7 MICROSOFT CORPORATION,
8 Petitioner,
9 vs.
10 DIRECTSTREAM, LLC,
11 Patent Owner.

12 -----
13 Case IPR2018-01599 (Patent 6,076,152)
14 Case IPR2018-01600 (Patent 6,247,110)
15 -----

16
17 VIDEO DEPOSITION OF STEPHEN M. TRIMBERGER, Ph.D.
18 Washington, D.C.
19 Friday, June 7, 2019, 9:00 a.m.
20
21
22
23

24 Job Number 161500

25 Reported by: Laurie Donovan, RFR, CRR, CSR

TSG Reporting - Worldwide 877-702-9580

20 A Could be. I don't recall any specific
21 cases, but talking to customers, reading
22 publications, this is, this is all part of that
23 activity.

24 Q Okay. So then what was your
25 understanding back at the time, pre-December '97,

Page 130

1 STEPHEN M. TRIMBERGER, Ph.D.
2 of the ways in which folks in the high performance
3 computing world viewed the advantages and
4 disadvantages of FPGAs?

5 A So again I think we need to make a
6 distinction here, and it's, and it's very relevant
7 to, to the time frame we're talking about. In
8 the, in the 1990s, people in high performance
9 computing, people who were doing, seriously doing
10 high performance computing did not use FPGAs.

11 So if you're saying, well, you know, how
12 would the people building the fastest computers
13 have viewed FPGAs, they would have viewed them --
14 I would -- my, my understanding is they viewed
15 them largely irrelevant, and there's evidence for
16 that, that there just are not FPGAs in the high
17 performance computers of the day except in among
18 the academic community, and that's the, the
19 community we're, we're seeing exposure to, and a
20 few, a few targeted research sites.

Secondary Considerations

Page 1

1
2 UNITED STATES PATENT AND TRADEMARK OFFICE
3 BEFORE THE PATENT TRIAL AND APPEAL BOARD
4
5 MICROSOFT CORPORATION,)
6)
7 Petitioner,)
8) IPR 2018-01594
9)
10 vs.) Patent 6,434,687
11)
12 SAINT REGIS MOHAWK TRIBE,)
13)
14 Patent Owner.)
15 -----x

14 VIDEOTAPED DEPOSITION OF HAROLD S. STONE, Ph.D.
15 Bellevue, Washington
16 Thursday, May 30, 2019
17
18
19
20
21
22
23 Reported by:
24 Connie Recob, CCR 2631, RMR, CRR
25 JOB NO. 160990

TSG Reporting - Worldwide 877-702-9580

PATENT OWNER DIRECTSTREAM, LLC
EX. 2066, p. 1

Page 168

1 HAROLD S. STONE, Ph.D.
2 I want to have you do is kind of walk through
3 with me: Imagine this is a person of ordinary
4 skill in the art at the time of the invention
5 who's working at Google, let's say, so they
6 have familiarity with Google's web server farm,
7 correct?
8 A. Yes.
9 Q. That person would also be aware of
10 other I guess computer architectures, correct?
11 A. Yes.
12 Q. And is it your testimony that one of
13 those would be an FPGA system?
14 A. Yes, because I have art that shows
15 that was there.
16 Q. And that -- would that person of
17 ordinary skill in the art at the time of the
18 invention also be aware of the limitations with
19 FPGAs?
20 A. All limitations, or some or -- I
21 mean, I -- he would be aware of FPGAs, he'd be
22 aware of the literature. That -- that's what
23 he's -- that's what that person skilled in the
24 art would know.
25 Q. And that -- that knowledge of the

Page 179

1 HAROLD S. STONE, Ph.D.
2 THE WITNESS: The question is --
3 what, what is your question now?
4 BY MR. HSU:
5 Q. Okay. So let me try again.
6 Would a person of ordinary skill in
7 the art have known that FPGAs come with
8 architectural tradeoffs?
9 A. I'm -- I'm -- I'm -- I'm certain
10 that there's some tradeoffs that they would
11 know about. I don't know if specifically
12 these, but the answer is, yes, they would know
13 there were tradeoffs.

Page 169

1 HAROLD S. STONE, Ph.D.
2 FPGA would include disadvantages, right?
3 A. If it's in the literature, if
4 it's -- if it's part of the art, yes.
5 MR. HSU: Okay. All right. Let me
6 introduce another exhibit here.
7 (Exhibit No. 2053 marked
8 for identification.)
9 BY MR. HSU:
10 Q. So this is marked Exhibit 2053. And
11 do you see that on your screen?
12 A. I see that.
13 Q. Okay. So this is an article titled:
14 "Improving Usability of FPGA-Based
15 Reconfigurable Computers Through Operating
16 System Support," and has two authors, So and
17 Brodersen.
18 Do you see that?
19 A. I see that.
20 Q. And this is from, it looks like the
21 University of California at Berkeley.
22 So in the abstract, that first
23 paragraph, in the last sentence, do you see
24 where it says: "The lack of common, intuitive
25 operating system support, however, hinders

Page 197

1 HAROLD S. STONE, Ph.D.
2 THE WITNESS: I -- I believe that is
3 the case, and the -- let me just say yes on
4 that.
5 BY MR. HSU:
6 Q. And do you agree that in 2001, at
7 the time of the invention, the FPGAs that were
8 available did not have the capability of
9 performing floating-point operations?
10 A. I don't know yes or no on that. I'd
11 have to look.

Secondary Considerations

**Failure of Others /
Commercial Success / Recognition / Praise**

Secondary Considerations

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORPORATION,

Petitioner,

v.

DIRECTSTREAM, LLC,

Patent Owner.

IPR2018-01594 (Patent 6,434,687 B1)

IPR2018-01599 (Patent 6,076,152)

IPR2018-01600 (Patent 6,247,110 B1)

IPR2018-01601 (Patent 7,225,324 B2)

IPR2018-01602 (Patent 7,225,324 B2)

IPR2018-01603 (Patent 7,225,324 B2)

IPR2018-01604 (Patent 7,421,524 B2)

IPR2018-01605 (Patent 7,620,800 B2)

IPR2018-01606 (Patent 7,620,800 B2)

IPR2018-01607 (Patent 7,620,800 B2)

DECLARATION OF DR. TAREK EL-GHAZAWI

PATENT OWNER DIRECTSTREAM, LLC
EX. 2164, p. 1

24. I co-authored various papers on the topic and I am attaching to this declaration two that I co-authored with Dr. Buell (USC, and Project Manager of the Splash 2), Dr. Gaj (GMU) and Dr. Kindratenko (UIUC). The first of those two papers [BUEL2007] is an introduction to the guest editors where the IEEE Computer Magazine. This simply establishes that even in 2007 this was a very hot topic and all contributions were valued. The article highlights the scalable SRC crossbar switch and indicates that when it comes to application development using the available commercial systems at the time, SRC provided a semi-integrated microprocessor-FPGA solution and where the hardware was side was developed still in C or FORTRAN dialect, while the rest of the vendors did not have an integrated system level solution and only addressed the FPGA side using third party FPGA only tools. The second paper, introduced a year later also by the four co-authors [EL-GH2008] attempts to survey state of the art and assess where the industry and research stood at that time.

Secondary Considerations

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORPORATION,

Petitioner,

v.

DIRECTSTREAM, LLC,

Patent Owner.

IPR2018-01594 (Patent 6,434,687 B1)

IPR2018-01599 (Patent 6,076,152)

IPR2018-01600 (Patent 6,247,110 B1)

IPR2018-01601 (Patent 7,225,324 B2)

IPR2018-01602 (Patent 7,225,324 B2)

IPR2018-01603 (Patent 7,225,324 B2)

IPR2018-01604 (Patent 7,421,524 B2)

IPR2018-01605 (Patent 7,620,800 B2)

IPR2018-01606 (Patent 7,620,800 B2)

IPR2018-01607 (Patent 7,620,800 B2)

DECLARATION OF DR. TAREK EL-GHAZAWI

PATENT OWNER DIRECTSTREAM, LLC
EX. 2164, p. 1

momentum in the High-Performance Reconfigurable Computing has risen then and traditional HPC companies like Silicon Graphics and Cray decided to join in the movement. The Cray solution was to buy an existing FPGA company and leverage their products, while the SGI Strategy was to build an FPGA board and integrated it their Altix product line. SRC commercial systems were ahead of the crowd for two reasons. First, for SRC, this was a core strategic product and nearly defining all what the company was focused on, and SRC as a commercial product for a high-performance reconfigurable computer was the first well integrated such system. Secondly, having come from a Vector Processing background (namely the Cray culture), SRC was very well positioned to bring to the table many ideas inspired by vector processing both on the side of code and architecture optimization. Examples of these are the development of the crossbar switch. The crossbar switch did not only connect the FPGA boards in a peer-to-peer fashion, but also provided connections to the modules of an interleaved shared memory subsystem much like a Cray vector architecture, and also added the microprocessor boards providing a great deal of flexibility. The support of such a large interleaved shared memory enables tackling much larger problems that go beyond embarrassingly parallel ones where all data can be kept local to the processor most of the time. The other high-performance computing companies on the other hand shortly abandoned their FPGA accelerated products and SRC continued down this path.

Secondary Considerations

RESEARCH FEATURE

The Promise of High-Performance Reconfigurable Computing

Tarek El-Ghazawi, Esam El-Araby, and Miaoqing Huang, George Washington University
Kris Gaj, George Mason University
Volodymyr Kindratenko, University of Illinois at Urbana-Champaign
Duncan Buell, University of South Carolina

Several high-performance computers now use field-programmable gate array coprocessors. The authors describe the two major contemporary HPRC architectures, the pros and cons of each using representative applications from remote sensing, dynamics, bioinformatics, and cryptanalysis.

In the past few years, high-performance computing vendors have introduced many systems containing both microprocessors and field-programmable gate arrays. Three such systems—the Cray XD1, the SRC-6, and the SGI Altix/RASC—are parallel computers that resemble modern HPC architectures, with added FPGA chips. Two of these machines, the Cray XD1 and SGI Altix, also function as traditional HPCs without the reconfigurable chips. In addition, several Beowulf cluster installations contain one or more FPGA cards per node, such as HPTI's reconfigurable cluster from the Air Force Research Laboratory.

In all of these architectures, the FPGAs serve as coprocessors to the microprocessors. The main application executes on the microprocessors, while the FPGAs handle kernels that have a long execution time but lend themselves to hardware implementations. Such kernels are typically data-parallel overlapped computations that can be efficiently implemented as fine-grained architectures, such as single-instruction, multiple-data (SIMD) engines, pipelines, or systolic arrays, to name a few.

Figure 1 shows that a transfer of control can occur during execution of the application on the microprocessor, in which case the system invokes an appropriate architecture in a reconfigurable processor to execute the target operation. To do so, the reconfigurable pro-

cessor can configure or reconfigure itself, while the system's other processors continue their computations. This feature is usually called reconfiguration.

From an application development perspective, users can create the hardware description languages such as VHDL or Verilog. High-level systems allow the use of high-level languages such as C and C++. SRC Computers' Carte C and C Accelerated Technologies' Impu Mitronics, and Celoxica's Hana high-level graphical programming languages such as Annapolis Micro System Systems' Viva, Xilinx System Generator, and Altera's Reconfigurable Computing Tool.

Readers should consult the special issue on high-performance computing for a good overview of microprocessor-based architectures and application-development tools and frameworks, and applications.

HPRC ARCHITECTURAL TAXONOMY

Many early HPRC systems, such as the SRC-6E and the Starbridge Hypercomputer, can be seen as attached coprocessors. These systems were designed around one node of microprocessors and another of FPGAs. The

In the past few years, high-performance computing vendors have introduced many systems containing both microprocessors and field-programmable gate arrays. Three such systems—the Cray XD1, the SRC-6, and the SGI Altix/RASC—are parallel computers that resemble modern HPC architectures, with added FPGA chips. Two of these machines, the Cray XD1 and SGI Altix, also function as traditional

0018-9162/08/0225-00 © 2008 IEEE

Published by the IEEE Computer Society

February 2008 59

PATENT OWNER DIRECTSTREAM, LLC
EX. 2165, p. 2

Secondary Considerations

RESEARCH FEATURE

The Promise of High-Performance Reconfigurable Computing

Tarek El-Ghazawi, Esam El-Araby, and Miaoqing Huang, George Washington University
Kris Gaj, George Mason University
Volodymyr Kindratenko, University of Illinois at Urbana-Champaign
Duncan Buell, University of South Carolina

Several high-performance computers now use field-programmable gate arrays as reconfigurable coprocessors. The authors describe the two major contemporary HPRC architectures and explore the pros and cons of each using representative applications from remote sensing, molecular dynamics, bioinformatics, and cryptanalysis.

In the past few years, high-performance computing vendors have introduced many systems containing both microprocessors and field-programmable gate arrays. Three such systems—the Cray XD1, the SRC-6, and the SGI Altix/RASC—are parallel computers that resemble modern HPC architectures, with added FPGA chips. Two of these machines, the Cray XD1 and SGI Altix, also function as traditional HPCs without the reconfigurable chips. In addition, several Beowulf cluster installations contain one or more FPGA cards per node, such as HPTI's reconfigurable cluster from the Air Force Research Laboratory.

In all of these architectures, the FPGAs serve as coprocessors to the microprocessors. The main application executes on the microprocessors, while the FPGAs handle kernels that have a long execution time but lend themselves to hardware implementations. Such kernels are typically data-parallel overlapped computations that can be efficiently implemented as fine-grained architectures, such as single-instruction, multiple-data (SIMD) engines, pipelines, or systolic arrays, to name a few.

Figure 1 shows that a transfer of control can occur during execution of the application on the microprocessor, in which case the system invokes an appropriate architecture in a reconfigurable processor to execute the target operation. To do so, the reconfigurable pro-

cessor can configure or reconfigure the FPGA "on the fly," while the system's other processors perform computations. This feature is usually referred to as runtime reconfiguration.¹

From an application development perspective, developers can create the hardware kernel using hardware description languages such as VHDL and Verilog. Other systems allow the use of high-level languages such as SRC Computers' Carte C and Carte Fortran, Impulse Accelerated Technologies' Impulse C, Mittron C from Mittronics, and Celoxica's Handel-C. There are also high-level graphical programming development tools such as Annapolis Micro Systems' CoreFire, Starbridge Systems' Viva, Xilinx System Generator, and DSPlogic's Reconfigurable Computing Toolbox.

Readers should consult *Computer's* March 2007 special issue on high-performance reconfigurable computing for a good overview of modern HPRC systems, application-development tools and frameworks, and applications.

HPRC ARCHITECTURAL TAXONOMY

Many early HPRC systems, such as the SRC-6E and the Starbridge Hypercomputer, can be seen as attached processors. These systems were designed around one node of microprocessors and another of FPGAs. The

0018-9162/08/25.00 © 2008 IEEE

Published by the IEEE Computer Society

February 2008 59

PATENT OWNER DIRECTSTREAM, LLC
EX. 2165, p. 2

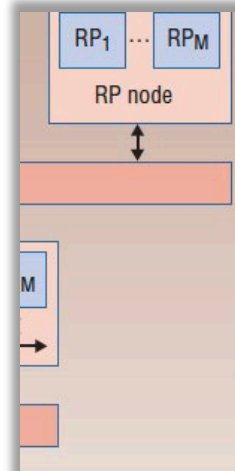


Figure 1: (a) uniform node architecture; (b) non-uniform systems (NNUs).

nodes.

A representative example of the UNNS is the SRC-6/SRC-7, which consists of one or more general-purpose microprocessor subsystems, one or more MAP reconfigurable subsystems, and global common memory (GCM) nodes of shared memory space. These subsystems are interconnected through a Hi-Bar switch communication layer. The microprocessor boards each include two 2.8-GHz Intel Xeon microprocessors and are connected to the Hi-Bar switch through a SNAP interface. The SNAP card plugs into the dual in-line memory module slot on the microprocessor motherboard to provide higher

ble
sor
uch
vell
res
e in
ous
ins
ro-
ern
orm
orm

data transfer rates between the boards than the less efficient but common peripheral component interconnect (PCI) solution. The sustained transfer rate between a microprocessor board and the MAP processors is 1,400 Mbytes per second.

The MAP Series C processor consists of one control FPGA and two user FPGAs, all Xilinx Virtex II-6000-4s. Additionally, each MAP unit contains six interleaved banks of onboard memory (OBM) with a total capacity of 24 Mbytes. The maximum aggregate data transfer rate among all FPGAs and OBM is 4,800 MBps. The user FPGAs are configured such that one is in master mode and the other is in slave mode. A bridge port directly connects a MAP's two FPGAs. Further, MAP processors can be connected via a chain port to create an FPGA array.

Secondary Considerations

RESEARCH FEATURE

The Promise of High-Performance Reconfigurable Co

Tarek El-Ghazawi, Esam El-Araby, and Miaoqing Huang, George Mason University
 Kris Gaj, George Mason University
 Volodymyr Kindratenko, University of Illinois at Urbana-Champaign
 Duncan Buell, University of South Carolina

Several high-performance computers now use field-programmable coprocessors. The authors describe the two major contemporary architectures, the pros and cons of each using representative applications from dynamics, bioinformatics, and cryptanalysis.

In the past few years, high-performance computing vendors have introduced many systems containing both microprocessors and field-programmable gate arrays. Three such systems—the Cray XD1, the SRC-6, and the SGI Altix/RASC—are parallel computers that resemble modern HPC architectures, with added FPGA chips. Two of these machines, the Cray XD1 and SGI Altix, also function as traditional HPCs without the reconfigurable chips. In addition, several Beowulf cluster installations contain one or more FPGA cards per node, such as HPTI's reconfigurable cluster from the Air Force Research Laboratory.

In all of these architectures, the FPGAs serve as coprocessors to the microprocessors. The main application executes on the microprocessors, while the FPGAs handle kernels that have a long execution time but lend themselves to hardware implementations. Such kernels are typically data-parallel overlapped computations that can be efficiently implemented as fine-grained architectures, such as single-instruction, multiple-data (SIMD) engines, pipelines, or systolic arrays, to name a few.

Figure 1 shows that a transfer of control can occur during execution of the application on the microprocessor, in which case the system invokes an appropriate architecture in a reconfigurable processor to execute the target operation. To do so, the reconfigurable pro-

cessor can configure, while the system reconfigures. This is

From an application developer's perspective, creating a description language for systems allow the SRC Computers' Accelerated Technology Mitronics, and C

high-level graphics such as Annapolis Systems' Viva, Xilinx Reconfigurable Co. Readers should special issue on high computing for a good application-developer applications.

HPC ARCHITECTURE

Many early HPCs the Starbridge Hy processors. These node of micropro-

			Expected		Measured		
			Throughput (GCUPS)	Speedup	Throughput (GCUPS)	Speedup	
FASTA (ssearch34)	Opteron 2.4 GHz	DNA	NA	NA	0.065	1	
		Protein	NA	NA	0.130	1	
SRC-6 100 MHz (32x1)	DNA	1 Engine/chip	3.2	49.2x	3.19 → 12.2 1 → 4 chips	49 → 188 1 → 4 chips	
		4 Engines/chip	12.8	197x	12.4 → 42.7 1 → 4 chips	191 → 656 1 → 4 chips	
		8 Engines/chip	25.6	394x	24.1 → 74 1 → 4 chips	371 → 1,138 1 → 4 chips	
	Protein		3.2	24.6x	3.12 → 11.7 1 → 4 chips	24 → 90 1 → 4 chips	
	XD1 200 MHz (32x1)	DNA	1 Engine/chip	6.4	98x	5.9 → 32 1 → 6 chips	91 → 492 1 → 6 chips
			4 Engines/chip	25.6	394x	23.3 → 120.7 1 → 6 chips	359 → 1,857 1 → 6 chips
8 Engines/chip			51.2	788x	45.2 → 181.6 1 → 6 chips	695 → 2,794 1 → 6 chips	
Protein		6.4	49x	5.9 → 34 1 → 6 chips	45 → 262 1 → 6 chips		

Figure 4. DNA and protein sequencing on the SRC-6 and Cray XD1 versus the open source FASTA program. An FPGA with one engine produced a 91x speedup, while eight cores on the same chip collectively achieved a 695x speedup.

PATENT OWNER DIRECTSTREAM, LLC
 EX. 2165, p. 2

Secondary Considerations

QUEST EDITORS' INTRODUCTION

Reconfigurable Computing

High-Performance Reconfigurable Computing

Duncan Buell, University of South Carolina
Tarek El-Ghazawi, George Washington University
Kris Gaj, George Mason University
Volodymyr Kindratenko, University of Illinois at Urbana-Champaign

High-performance reconfigurable computers have the potential to exploit coarse-grained functional parallelism as well as fine-grained instruction-level parallelism through direct hardware execution on FPGAs.

High-performance reconfigurable computers (HPRCs)^{1,2} based on conventional processors and field-programmable gate arrays (FPGAs)³ have been gaining the attention of the high-performance computing community in the past few years.⁴ These synergistic systems have the potential to exploit coarse-grained functional parallelism as well as fine-grained instruction-level parallelism through direct hardware execution on FPGAs.

HPRCs, also known as reconfigurable supercomputers, have shown orders-of-magnitude improvement in performance, power, size, and cost over conventional high-performance computers (HPCs) in some compute-intensive integer applications. However, they still have not achieved high performance gains in most general scientific applications. Programming HPRCs is still not straightforward and, depending on the programming tool, can range from designing hardware to software programming that requires substantial hardware knowledge.

The development of HPRCs has made substantial progress in the past several years, and nearly all major high-performance computing vendors now have HPRC product lines. This reflects a clear belief that HPRCs

108-0162/12/\$12.00 © 2012 IEEE

Published by the IEEE Computer Society

March 2012 23

PATENT OWNER DIRECTSTREAM, LLC
EX. 2166, p. 2



have tremendous potential and that resolving all remaining issues is just a matter of time.

This special issue will shed some light on the state of the field of high-performance reconfigurable computing.

WHAT ARE HIGH-PERFORMANCE RECONFIGURABLE COMPUTERS?

HPRCs are parallel computing systems that contain multiple microprocessors and multiple FPGAs. In current settings, the design uses FPGAs as coprocessors that are deployed to execute the small portion of the application that takes most of the time—under the 10-90 rule, the 10 percent of code that takes 90 percent of the execution time. FPGAs can certainly accomplish this when computations lend themselves to implementation in hardware, subject to the limitations of the current FPGA chip architectures and the overall system data transfer constraints.

In theory, any hardware reconfigurable devices that change their configurations under the control of a program can replace the FPGAs to satisfy the same key concepts behind this class of architectures. FPGAs, however, are the currently available technology that provides the most desirable level of hardware reconfigurability. Xilinx, followed by Altera, dominates the FPGA market, but new startups are also beginning to enter this market.

FPGAs are based on SRAM, but they vary in structure. Figure A in the "FPGA Architecture" sidebar shows an FPGA's internal structure based on the Xilinx architecture style. The configurable logic block (CLB) is the basic building block for creating logic. It includes RAM used as a lookup table and flip-flops for buffering, as well as multiplexers and carry logic. A side-by-side 2D array of switching matrices for programmable routing connects the 2D array of CLBs.

PROGRESS IN SYSTEM HARDWARE AND PROGRAMMING SOFTWARE

During the past few years, many hardware systems have begun to resemble parallel computers. When such systems originally appeared, they were not designed to be scalable—they were merely a single board of one or more FPGA devices connected to a single board of one or more microprocessors via the microprocessor bus or the memory interface.

The recent SRC-6 and SRC-7 parallel architectures from SRC Computers use a crossbar switch that can be stacked for further scalability. In addition, traditional high-performance computing vendors—specifically, Silicon Graphics Inc. (SGI), Cray, and LinuxNeworks—have incorporated FPGAs into their parallel architec-

tures. In addition to the SRC-7, models of such HPC systems include the SGI RASC RC100 and the Cray XD1 and XT4. The Linux Network work focuses on the design of the acceleration boards and on coupling them with PC nodes for constructing clusters.

On the software side, SRC Computers provides a semi-integrated solution that addresses the hardware (FPGA) and software (microprocessor) sides of the application separately. The hardware side is expressed using Carte C or Carte Fortran as a separate function, compiled separately and linked to the compiled C (or Fortran) software side to form one application.

Other hardware vendors use a third-party software tool, such as Impulse C, Handel-C, Mitron C, or DSPlogic's RC Toolbox. However, these tools handle only the FPGA side of the application, and each machine has its own application interface to call those functions. At present, Mitron C and Handel-C support the SGI RASC, while Mitron C, Impulse C, and RC Toolbox support the Cray XD1. Only a library-based parallel tool such as the message-passing interface can handle scaling an application beyond one node in a parallel system.

RESEARCH CHALLENGES AND THE EVOLVING HPRC COMMUNITY

FPGAs were first introduced as glue logic and eventually became popular in embedded systems. When FPGAs were applied to computing, they were introduced as a back-end processing engine that plugs into a CPU bus. The CPU in this case did not participate in the computation, but only served as the front end (host) to facilitate working with the FPGA.

The limitations of each of these scenarios left many issues that have not been explored, yet they are of great importance to HPRC and the scientific applications it targets. These issues include the need for programming tools that address the overall parallel architecture. Such tools must be able to exploit the synergy between hardware and software execution and should be able to understand and exploit the multiple granularities and localities in such architectures.

The need for parallel and reconfigurable performance profiling and debugging tools also must be addressed. With the multiplicity of resources, operating system support and middleware layers are needed to shield users from having to deal with the hardware's intricate details. Further, application-portability issues should be thoroughly investigated. In addition, new chip architectures that can address the floating-point requirements of scientific applications should be explored. Portable libraries that can support scientific applications must be

HPRCs are parallel computing systems that contain multiple microprocessors and multiple FPGAs.

24 Computer

PATENT OWNER DIRECTSTREAM, LLC
EX. 2166, p. 3

Motions to Exclude

Disputed Exhibits

TABLE OF CONTENTS

I.	STATEMENT OF OBJECTIONS	1
II.	LEGAL STANDARDS	2
III.	ARGUMENT	4
	A. Uses of the Evidence	4
	B. Exhibits 1074, 1077 and 1079 (Technical Documents)	6
	C. Exhibit 1076 (Dr. Stone Reply Declaration)	8
	D. Exhibit 1075 (Dr. Homayoun Deposition Transcript)	9
	E. Exhibit 1078 (Dr. El-Ghazawi Deposition Transcript)	11
IV.	CONCLUSION	12
V.	CERTIFICATE OF SERVICE	14
VI.	CERTIFICATE OF PAGE COUNT	15

TABLE OF CONTENTS

I.	STATEMENT OF OBJECTIONS	1
II.	LEGAL STANDARDS	2
III.	ARGUMENT	4
	A. Uses of the Evidence	4
	B. Exhibits 1074, 1077 and 1079 (Technical Documents)	6
	C. Exhibit 1076 (Dr. Stone Reply Declaration)	8
	D. Exhibit 1075 (Dr. Homayoun Deposition Transcript)	9
	E. Exhibit 1078 (Dr. El-Ghazawi Deposition Transcript)	11
IV.	CONCLUSION	12
V.	CERTIFICATE OF SERVICE	14
VI.	CERTIFICATE OF PAGE COUNT	15

IPR2018-01601, -01602, -01603

U.S. Patent No. 7,225,324

TABLE OF CONTENTS

I.	Introduction	1
II.	Exhibit 2100	1
III.	Exhibits 2066, 2076, and 2092	6
IV.	Exhibits 2067-2075, 2077, 2079-2099, 2101-2103, 2105-2106, 2109-2110, 2112-2133, 2139-2151, 2155, 2161-2163, and 2168	7
V.	Exhibit 2168	8
VI.	Exhibit 2111	8
VII.	Patent Owner Response	9
VIII.	Conclusion	9

IPR2018-01605, -01606, -01607

U.S. Patent No. 7,680,800

TABLE OF CONTENTS

I.	Introduction	1
II.	Exhibit 2101	1
III.	Exhibits 2065, 2075, and 2091	6
IV.	Exhibits 2066-2074, 2076, 2078-2099, 2102-2104, 2106-2107, 2110, 2113-2134, 2140-2152, 2156, 2163, 2165, and 2170	7
V.	Exhibit 2170	8
VI.	Exhibit 2112	8
VII.	Patent Owner Response	9
VIII.	Conclusion	9